МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Алгоритмічні і програмні компоненти програмного комплексу для проектування нейронних мереж»

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

Виконав: студент групи МгІТ-21

Гучок Олег Вікторович

Науковий керівник к.т.н., доц. Колиско М.І.

Рецензент

Київ 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій
Кафедра комп'ютерних наук
Рівень вищої освіти другий (магістерський)
Спеціальність 122 Комп'ютерні науки
Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
_____Володимир ЩЕРБАНЬ.
"____" _____2023 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Гучок Олегу Вікторовичу

1. Тема кваліфікаційної роботи Алгоритмічні і програмні компоненти програмного комплексу для проектування нейронних мереж, науковий керівник роботи Колиско Мар'яна Ігорівна, доц,к.т.н. затверджені наказом КНУТД від "_12_"вересня_2023 року №_210-уч_

2. Вихідні дані до роботи: Розробки кафедри комп'ютерних наук; рекомендована література, додатки.

3. Зміст дипломної роботи : Вступ; РОЗДІЛ 1 Постановка задачі; РОЗДІЛ 2 Проектування; РОЗДІЛ 3 Програмна реалізація; Висновки; Список літератури; ДОДАТОК А Окремі фрагменти програмного коду; ДОДАТОК Б Презентація.

4. Дата видачі завдання_1 вересня 2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапу кваліфікаційної роботи	Орієнтовний термін виконання	Примітка про виконання
1	Вступ	*15.09.2023*	
2	Розділ 1. Постановка задачі	*20.09.2023*	
3	Розділ 2 Проектування	*30.09.2023*	
4	Розділ 3. Програмна реалізація	*10.10.2023*	
5	Висновки	*25.10.2023*	
6	Оформлення (чистовий варіант)	*1.11.2023*	
7	Подача кваліфікаційної роботи науковому керівнику для відгуку (за 14 днів до захисту)	*4.11.2023*	
8	Подача кваліфікаційної роботи для рецензування (за 12 днів до захисту)	*6.11.2023*	
9	Перевірка кваліфікаційної роботи на наявність ознак плагіату (за 10 днів до захисту)	*8.11.2023*	
10	Подання кваліфікаційної роботи на затвердження завідувачу кафедри (з 7 днів до захисту)	*10.11.2023*	

З завданням ознайомлений:

Студентка _____ Олег ГУЧОК

Науковий керівник _____ Мар’яна КОЛИСКО

АНОТАЦІЯ

Гучок О.В. Алгоритмічні і програмні компоненти програмного комплексу для проектування нейронних мереж. – Рукопис.

Кваліфікаційна магістерська робота за спеціальністю 122 – «Комп'ютерні науки ». – Київський національний університет технологій та дизайну, Київ, 2023 рік.

Штучні нейронні мережі знаходять застосування у наступних сферах: класифікація та розпізнавання образів, системи асоціативної пам'яті, компресія даних, оптимізаційні задачі, теорія керування, розробка нейрокомп'ютерів, наближення функцій з високою точністю, екстраполяція та прогнозування. Метою кваліфікаційної роботи є розробка програмного додатку для побудови штучної нейронної мережі який можна було б використовувати для широкого спектру задач.

У роботі наведено перелік архітектури та методів навчання нейронних мереж. Розроблено алгоритм та архітектуру програми.

Випускна кваліфікаційна роботи займає___ сторінок, включає ___ малюнків, ___ таблиць, ____ додатка і складається з 3 розділів.

В першому розділі зроблено аналіз предметної області. Було розглянуто сучасний стан машинного навчання, зокрема глибинного навчання заснованого на застосуванні нейронних мереж.

В другому розділі розглянуто математичні моделі та алгоритми функціонування штучних нейронних мереж та їх складових.

У третьому розділі описано складові програмного додатку та його застосування для побудови і демонстрації роботи нейронної мережі .

Розроблений програмний засіб можна буде використовувати для проектування нейронних мереж на різних операційних системах та апаратних платформах

Ключові слова: нейронні мережі, нейрони, машинне навчання.

ABSTRACT

Guchok O.V. Algorithmic and software components of the software complex for designing neural networks. - Manuscript.

Qualifying master's thesis in specialty 122 - "Computer science". - Kyiv National University of Technology and Design, Kyiv, 2023.

Artificial neural networks are used in the following areas: classification and pattern recognition, associative memory systems, data compression, optimization problems, control theory, development of neurocomputers, approximation of functions with high accuracy, extrapolation and forecasting.

The goal of the qualification work is to develop a software application for building an artificial neural network that could be used for a wide range of tasks.The work lists the architecture and learning methods of neural networks. The algorithm and architecture of the program have been developed.

The graduation thesis takes ___ pages, includes ___ figures, ___ tables, ____ appendix and consists of 3 sections.

In the first section, an analysis of the subject area is made. The current state of machine learning, in particular deep learning based on the application of neural networks, was considered.

In the second chapter, mathematical models and algorithms of the functioning of artificial neural networks and their components are considered.

The third section describes the components of the software application and its use for building and demonstrating the operation of a neural network.

The developed software can be used to design neural networks on various operating systems and hardware platforms

Keywords: neural networks, neuron, machine learning.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

МН – (Машинне навчання, machine learning, ML) - набір алгоритмів, які утворюють систему, що навчається на власному досвіді.

КМН (Класичне машинне навчання, або Classical Machine Learning) - побудоване на класичних статистичних алгоритмах і вирішує питання, пов'язані з прийняттям рішень на основі даних

ШНМ (штучна нейронна мережа) – обчислювальна нелінійна модель, в основі якої є нейронна структура мозку. Вона здатна навчатися виконанню задач класифікації, передбачення, прийняття рішень та візуалізації.

LSTM (Long Short-Term Memory) – мережа довгої короткострокової пам'яті

МЗПП – метод зворотного поширення помилки.

ФА – функція активації.

ПШ – прихований (проміжний) шар.

ВхШ – вхідний шар.

ВдШ – вихідний шар

ВСТУП

Нейромережеві технології проникають у все більшу кількість інформаційних систем й використовуються для покращення користувацького досвіду під час роботи із системою (адаптованість системи під функції, які користувач використовує більше, стилізація відповідно до смаків користувача, тощо), виконання евристичних завдань у конкретній предметній області (розпізнавання обличь на фотографії, знаходження закономірностей у великих обсягах даних, штучний інтелект у іграх, очищення шуму в звуковому сигналі, тощо).

Перші кроки в області штучних нейронних мереж були зроблені в 1943 р. В. Мак-Калохом (W.MCculloch) та В. Пітсом (W.Pitts). Вони показали, що за допомогою порогових нейронних елементів можна реалізувати обчислення будь-яких логічних функцій. У 1957-1962 рр. Ф.Розенблатом була запропонована і досліджувана модель нейронної мережі, яку він назвав персептроном. Результати досліджень він узагальнив в книзі "Принципи нейродинаміки". В 1969р. М.Мінський і С. Пайперт опублікували монографію "Персептрони", де був проведений математичний аналіз персептрона, і показані обмеження, властиві йому. Висновки були досить песимістичними, тож це зіграло негативну роль в подальшому розвитку досліджень в області нейронних мереж. Роботи в цій області були практично зупинені.

У 70-і роки з'явився ряд робіт в області асоціативної пам'яті. У 1982р. Д.Хопфілд дав аналіз стійкості нейронних мереж із зворотними зв'язками і запропонував використати їх для вирішення завдань оптимізації. Ряд авторів (Румельхарт, Хінтон, Вільямс) запропонували алгоритм зворотного поширення помилки, який став потужним засобом для навчання багатошарових нейронних мереж. У 1987р. під егідою IEEE (Institute of Electrical and Electronic Engineer's) проводиться перша міжнародна конференція в області нейронних мереж.

Нейрокомп'ютери ще не увійшли до нашого життя. Вони знаходяться десь на півдорозі. Проте росте число фірм, що вже продають вироби, що називаються нейрокомп'ютерами. Сотні лабораторій створюють нові варіанти машин шостого

покоління. Серед нейрокомп'ютерних спроб важливу роль грають моделі нейронних мереж і програмні імітатори нейрокомп'ютерів на РС - нейроімітатори. Вони виконують різні функції: учбові - дозволяють познайомитися з нейронними мережами, дослідницькі - дають можливість апробувати нові алгоритми і архітектуру, розважальні, - є цікавою інтелектуальною іграшкою, нарешті, нейроімітатори на РС стали конкурентоздатним забезпеченням для вирішення ряду прикладних завдань

Існують тільки загальні твердження про ефективність тієї чи іншої архітектури (мережі прямого поширення, рекурентні системи, системи з пам’яттю, тощо). Конкретна будова (кількість нейронів, функції належності, зв’язки між нейронами), має бути встановлена експериментальним шляхом на основі даних, які будуть аналізуватися. Такий підхід вимагає великої кількості спроб та не гарантує знаходження оптимальною чи субоптимальної структури. Тому створення засобу для автоматичного або напівавтоматичного проектування (із можливостями для подальших модифікацій) дозволить створювати нейронні мережі швидше і забезпечить їх кращу якість у порівнянні із спроектованими в повністю ручному режимі.

Мета роботи - створення прототипу засобу, який проектуватиме структуру нейронної мережі довільного типу відповідно до вказаної кількості вхідних сигналів для аналізу, очікуваної вихідної кількості сигналів, вказаного стартового шаблону та заданої стратегії побудови нейронної мережі. Для досягнення поставленої мети необхідно вирішити такі задачі: - проаналізувати структури даних, які дозволять представити структуру нейронних мереж; реалізувати нейромережевий рушій, який дозволить будувати нейронні мережі відповідно до створеної структури, виконувати обчислення із наборами вхідних даних та проводити навчання мережі; спроектувати засіб так щоб частини були незалежними одна від одної й існувала можливість змінювати реалізацію кожного окремого алгоритму.

РОЗДІЛ 1 ПОСТАНОВКА ЗАДАЧІ

Можна спостерігати що останнім часом поширюється інтерес до машинного навчання як до складової галузі штучного інтелекту. Машинне навчання досліджує та розробляє алгоритми, які наділяють комп'ютер здатністю навчатись на основі обробки і аналізу даних, «не будучи явно запрограмованими». Алгоритми ML мають здатність знаходити патерни, закономірності в процесі аналізу великих масивів даних. Це допомагає аналізувати поведінку, і шукати причинно наслідкові зв'язки в неочевидних людині речах. Алгоритми будують модель з тренувального набору даних, яка надалі здійснює передбачення на основі вхідних даних (Рис.1):

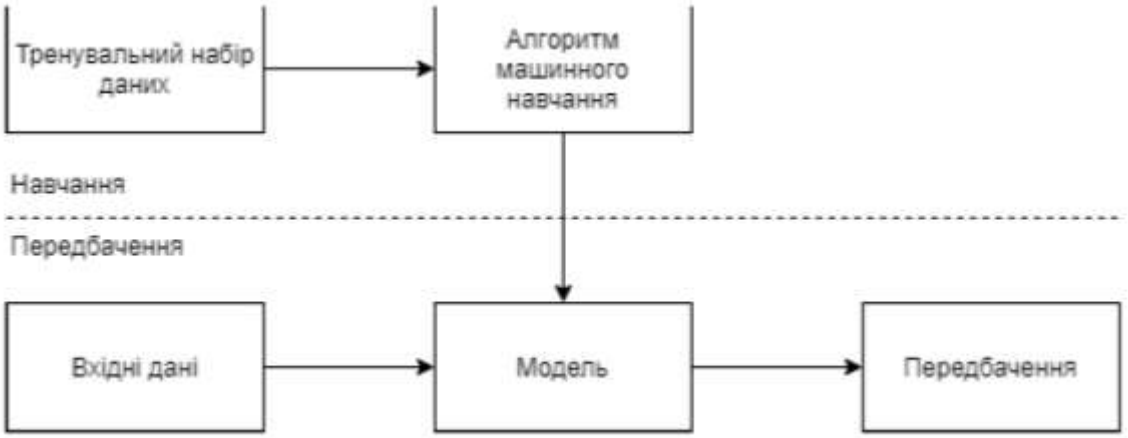

Рис. 1. Загальна схема роботи алгоритму машинного навчання

Машинне навчання засновано на трьох складових :

1) Дані – базова інформація. Сюди входять будь-які вибірки даних, роботі з якими потрібно навчити систему. Чим більше даних завантажити в систему, тим точніше і швидше вона працюватиме. Сутність даних безпосередньо залежить від завдання, яке стоїть перед машиною. Наприклад, щоб пошта навчилась відфільтровувати спам від листів, необхідні зразки-приклади. І чим більша їх вибірка, тим краще. Таким чином система вчиться сприймати конкретні слова – «Купити», «Додатковий дохід», «Заробляй вдома», «Схуднення» – як ознаки спаму. Вхідні дані для інших завдань будуть іншими. Щоб радити покупцеві товари, які можуть його зацікавити, потрібна історія здійснених ним покупок. Щоб передбачити зміну цін на ринку, потрібна історія цін. Найскладніша і

найбільш об'ємна частина роботи – збір цих самих даних. Існує два методи збору даних: вручну і автоматично. Ручний метод набагато повільніший, але при цьому точніший. Автоматичний же набагато швидший, але при цьому допускає більшу кількість помилок. Хороша вибірка даних дорогого коштує, адже саме вона відповідає за точність навчання. Дуже важливо не обмежувати збір даних людським мисленням, а надавати максимум розрізненої інформації, оскільки машина може побачити користь і взаємозв'язки там, де людина їх не помітить.

2) Ознаки (властивості, метрики, фічі, характеристики, features)– ця частина роботи проводиться в співпраці з замовником. Тут треба визначити ключові бізнес-потреби та вирішити, які саме характеристики та властивості повинна відстежувати система в результаті навчання. До властивостей належать ті характеристики, від яких безпосередньо залежить вихідний результат. Наприклад, у випадку з автомобілем такими фічами будуть пробіг, кількість циліндрів, максимально можлива швидкість. У випадку з покупцем: вік, стать, освіта, рівень доходу і т. п. У випадку з тваринами: порода, зріст, довжина від кінчика хвоста до носа, масть. Всі характеристики, розбиті на конкретні колонки, і є тими властивостями, які мережі використовують для видачі результату. Оскільки правильність характеристик безпосередньо впливає на результат, що ми отримаєм, то їх відбір займає набагато більше часу, ніж сам процес машинного навчання. Тут головне - не обмежувати набір характеристик, виходячи з особистої думки, щоб не спотворити машинне сприйняття. А разом із ним і кінцевий результат.

3) **Алгоритм** – вибір методу для вирішення поставленого бізнес-завдання. Це система послідовних операцій для вирішення певної задачі. Іншими словами – метод вирішення. Під кожну конкретну задачу можна підібрати окремий витончений алгоритм. Саме від обраного методу безпосередньо залежить швидкість і точність результату обробки вхідних даних. Бувають випадки, коли навіть ідеально написані алгоритми не допомагають вирішувати поставлені бізнес-завдання. Наприклад, якщо ви хочете збільшити кількість крос-продажів на сайті та впевнені, що для цього потрібно просто поліпшити алгоритм

рекомендації товарів. Але при цьому не знаєте, що ваші клієнти приходять за прямими посиланнями з пошуку і ігнорують поради щодо купівлі інших товарів, показані на сайті. Тому, перш ніж починати роботу, треба визначити реальну причину проблеми клієнта.

За ознакою наявності вчителя, навчання ділиться на: навчання з учителем (Supervised Learning), без вчителя (Unsupervised Learning) та з підкріпленням (Reinforcement Learning).

- Навчання з учителем застосовують коли потрібно навчити машину розпізнавати об'єкти або сигнали. Загальний принцип навчання з учителем це "дивись, ось це яблуко і це теж яблуко, і ось це теж яблуко "

- Навчання без вчителя використовує принцип "цей предмет такий же як інші". Алгоритми вивчають і фіксують подібності і можуть виявити відмінність та виконати виявлення аномалій, розпізнаючи те, що є незвичайним або несхожим.

- Навчання з підкріпленням використовують там, де перед машиною стоїть завдання - правильно виконати поставлені завдання у зовнішньому середовищі маючи множину можливих варіантів ді. Наприклад в комп'ютерних іграх, трейдингових операціях, або для безпілотної техніки.

За типом застосовуваних алгоритмів можна виділити два види:

1) класичне навчання - відомі і добре вивчені алгоритми навчання, що були розроблені в основному більше 50-ти років тому для статистичних бюро. Підходить, насамперед, під завдання роботи з даними: класифікація, кластеризація, регресія і т. п. Застосовують для прогнозування, сегментації клієнтів і так далі.

2) нейронні мережі і глибоке навчання - найбільш сучасний підхід до машинного навчання. Нейронні мережі застосовуються там, де потрібні розпізнавання або генерація зображень та відео, складні алгоритми управління або прийняття рішень, машинний переклад і подібні складні завдання.

Класичне машинне навчання, або Classical Machine Learning, будується на класичних статистичних алгоритмах і вирішує питання, пов'язані з прийняттям

рішень на основі даних. Його активно застосовують як в оф-лайн, так і в онлайн-маркетингу для прогнозування поведінки користувача і рекомендацій за інтересами. У класичному машинному навчанні з учителем програміст, який навчає систему, розмічає дані, приводить машині певні приклади та спостерігає за її прогресом. Завданнями, які вирішуються за допомогою навчання з учителем є, наприклад, класифікація **і** регресія**.**

Класифікація – найбільш популярна задача машинного навчання, вона подібна до того, як дитина вчиться визначати форму і розмір предметів, розкладаючи їх у роздільні купки. Завдання класифікації: передбачення категорії об'єкта або розподіл об'єктів згідно з визначеними і заданими наперед ознаками. Тобто, машина сортує дані за потрібними категоріями: одяг – за кольорами, сезонами або тканинами, книги – за жанрами, авторам, мовою написання, соуси – за ступенем гостроти, листи – за особистою чи корпоративною спрямованістю, спам-складовою і т. д. Додатковий продукт класифікації за заданими параметрами – можливість виділити все, що не вписується в стандартні класи. Наприклад, якщо мова про медицину, виділеним фрагментом може бути будь-яке відхилення від норми: потовщення, розрив, новоутворення, завищені чи занижені показники аналізів..

Регресія – це коли за заданим набором ознак треба спрогнозувати якусь цільову змінну. Завдання регресії: передбачення позиції на числовій прямій. Наприклад, завантаженість доріг залежно від часу доби та час на дорогу з пункту А в пункт Б залежно від заторів. Або яким буде обсяг ринку певних товарів через 2 роки. І навіть швидкість розвитку певної хвороби при загальних показниках здоров'я людини. Оскільки регресія запрограмована на роботу з числами, її вбудовують у різні обчислювальні системи, навіть у класичний Excel.

Цікаво що систему класифікації можна «довчити» і навчити вирішувати завдання регресії. На практиці це виглядає як розуміння не тільки класу об'єкта, але і його близькості до того чи іншого показника. Наприклад, яблуко свіже чи зіпсоване. А якщо ближче до зіпсованого, то на скільки відсотків.

Машинне навчання без учителя включає в себе наступні типи**:** кластеризація, узагальнення, пошук правил**.**

Кластерний аналіз (Data clustering) – задача розбиття заданої вибірки даних (об'єктів) таким чином, щоб кожен кластер складався зі схожих об'єктів, а об'єкти різних кластерів значно відрізнялися одне від одного. Завдання кластеризації – використовуючи всі наявні дані, передбачити відповідність об'єктів вибірки їхнім класам, сформувавши таким чином кластери. Кластеризацію застосовують для аналізу та пошуку ознак, за якими можна об'єднати об'єкти, стиснення даних і пошуку новизни (що не входить у жоден із кластерів)

Пошук асоціативних правил – метод, який активно використовується в маркетингу для вивчення поведінки покупця і складання типового шаблону покупок. Завдання пошуку асоціативних правил: знаходити закономірності в потоці даних. Наприклад, для аналізу патернів поведінки користувачів на веб-сайтах або для правильної розстановки товарів на полицях магазину. Адже недарма жуйки та батарейки розташовані в прикасовій зоні.

Одним із розділів машинного навчання є глибинне навчання (Deep Learning). Глибинне навчання займається розробкою алгоритмів для навчання на основі ознак даних із використанням багатошарових графів, що отримали назву штучних нейронних мереж. Архітектури глибинного навчання використовуються в таких прикладних задачах:

- розпізнавання мовлення та обробка природної мови;
- комп’ютерний зір;
- механізми фільтрації контенту в соціальних мережах;
- машинний переклад;
- пошукові алгоритми;
- біоінформатика;
- розробка ліків тощо.

Глибоке навчання (Deep learning) – метод машинного навчання, заснований, у першу чергу, на нейронних мережах, хоча можна застосовувати й інші методи. У

сучасній реальності практично у всьому, що стосується Deep Learning, використовують нейронні мережі.

Успіх глибокого навчання безпосередньо залежить від потужності техніки. На момент появи нейронних мереж потужності комп'ютерів були низькими, через що і самі мережі були досить слабкими. Саме тому в той час неможливо було створити велику кількість шарів нейронних мереж, а саме від кількості шарів залежать можливості мережі. Сучасний Deep Learning здатен упоратися з великими розмірами мереж. Для глибокого навчання використовують спеціальні фреймворки: Keras, Detectron, TensorFlow, PyTorch та інші.

Нейронна мережа за допомогою штучних нейронів моделює роботу людського мозку (нейронів), що вирішує певне завдання, самонавчається з урахуванням попереднього досвіду. І з кожним разом робить усе менше помилок. Нейромережі є одним із видів машинного навчання, а не окремим інструментом.

Нейронні мережі використовують практично у всіх завданнях, де людина намагається застосувати ШІ. Розглянемо буквально кілька прикладів:

Генерація віршів (RNN) - Цікавою особливістю даних мереж є їхнє вміння створювати власні унікальні слова, яких не було в словнику, на якому їх навчали.

Розпізнавання образів (CNN, Convolutional neural network**)** – одна з найвпливовіших інновацій в області комп'ютерного зору. Застосовуються скрізь, де необхідно розпізнати і/або класифікувати образи/обличчя.

Фотореалістичні зображення (GAN Generative Adversarial Nets) – можуть використовуватись, наприклад, у криміналістиці, коли потрібно створити фоторобот злочинця за описом, в дизайні – для створення предметів одягу або інтер'єру, виходячи з їхнього призначення.

Прийняття рішень і боти в іграх(DQN Deep Q Learning) – їх використовують для прийняття рішень ШІ на підстав аналізу поточної ситуації. Роботу таких нейронних мереж демонструють безпілотні автомобілі, трейдингові боти, чат-боти та ін. DQN лежить в основі машинного навчання типу "навчання без учителя".

Розглянемо особливості використання штучних нейронних мереж та окреслимо сучасні проблеми в їх проектуванні. Штучна нейронна мережа – математична модель, принцип роботи якої нагадує роботу мережі біологічних нейронів. Кожен шар представлений множиною вузлів – штучних нейронів, які видобувають ознаки все більшого рівня, поки останній шар не скомбінує ці ознаки, щоб зробити передбачення:

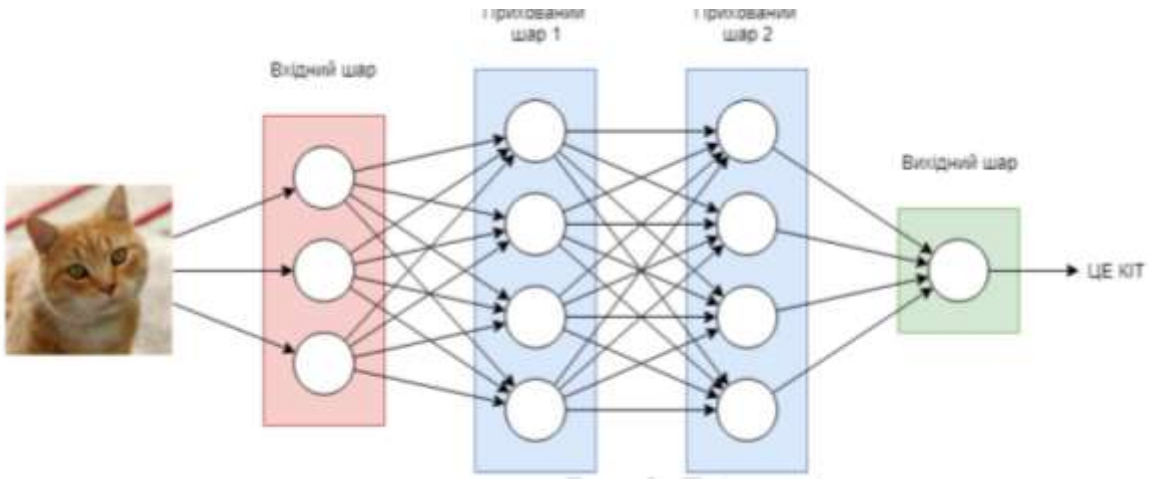

Рис. 2. Будова багатошарової нейронної мережі

Штучний нейрон – це модель, яка складається із шару вхідних сигналів , кожен із яких має свою вагу. Ваги відображають значимість кожного входу штучного нейрону. На основі вхідних активацій та їх ваг у вихідний шар обчислюється сумуюча функція, яка враховуючи вхідні сингали та їх ваги обчислює деяке значення. Далі результат сумуючої функції передається в активаційну функцію, яка в залежності від значення сумуючої функції обчислює вихідний результат нейрона.

Одним із спрощених моделей штучного нейрона є перцептрон. Він складається із вхідного та вихідного шару. На вхідному шарі приймається вектор вхідних сигналів *X*, кожен з яких має відповідну вагу. Активаційна функція обчислює результуюче значення нейрона, що передається далі:

$$f(x, w, b) = \begin{cases} 1 \text{ якщо } \sum_{i=1}^{n} w_i x_i + b > 0 \\ 0 \text{ якщо } \sum_{i=1}^{n} w_i x_i + b \leq 0 \end{cases},$$

де *x* – вектор вхідних активацій, *w* – вектор ваг, *b* – зміщення (bias), *f,x,w,b* – активаційна функція.

Розглянемо навчальний алгоритм для перцептрона:

```
perfect = False
 while NOT perfect:
 perfect = True
 for e in examples:
 if Predict(e) != Target(e):
 perfect = False
 if Predict(e) == 0:
 w = w + e
 if Predict(e) == 1:
 w = w - e
```

Вважаємо, що спочатку класифікація не ідеальна та здійснюємо цикл поки виконується умова такої неідеальності (поки змінна perfect рівна False). Для всіх значень прикладів масиву examples перевіряємо, якщо передбачене значення для даного прикладу не рівне цільовому значенню, то класифікація не ідеальна і алгоритм продовжує свій цикл. В цьому ж блоці, якщо невірно передбачене значення рівне 0, то до ваги додаємо значення поточного прикладу. Якщо ж невірно передбачене значення рівне 1, то відповідно від ваги віднімаємо значення поточного прикладу.

Скорочено даний алгоритм можна записати у вигляді:

$\Delta w_i = y - y^* x_i$

Де Δw_i зміна і-тої ваги після поточного прикладу, y – правильна вихідна активація перцептрона для даного прикладу (може бути рівна 0 або 1), y – поточна вихідна активація перцептрона для даного прикладу (може бути рівна 0 або 1), x_i – *i*-тий вхід на поточному прикладі.

Отже, завданням штучного нейрона є прийом вхідних сигналів, їх обробка та передача результату, що генерується активаційною функцією. Багаторівневі мережі, утворені зі штучних нейронів, здатні розв’язувати достатньо складні задачі.

Однією із необхідних умов ефективної роботи нейронної мережі є її паралельність обчислень. Послідовні обчислення центрального процесора персонального комп’ютера значно сповільнюють роботу нейронної мережі, особливо багатошарової з великою кількістю нейронів. Тому слід розглянути сучасні підходи при розробці нейронних мереж (як правило, дані підходи поєднуються):

- використання апаратних розподілених систем – нейрокомп’ютерів, графічних процесорів;

- створення паралельних обчислювальних систем на базі персональних комп’ютерів;

- використання хмарних сервісів

Широке використання графічних процесорів для паралельних обчислень та створення відповідних додатків стало можливе завдяки технології CUDA – платформі для паралельних обчислень на графічних процесорах від Nvidia. Для реалізації задач глибинного навчання розроблена бібліотека cuDNN, що забезпечує такі стандартні процедури нейронних мереж як пряме поширення, метод зворотного поширення помилки, об’єднуючий (pooling), нормуючий (normalization) та активаційний (activation) шари. Дана бібліотека підтримує такі фреймворки як TensorFlow, Caffe2, MATLAB, Microsoft Cognitive Toolkit, Theano, PyTorch.

Використання графічних процесорів з підтримкою технології CUDA для проведення досліджень з паралельних обчислень та зокрема глибинного навчання є хорошим рішенням. Проте доцільно звернути увагу на альтернативну можливість, наприклад, використання хмарних сервісів. Лідерами в напрямку хмарних технологій, зокрема і в проектуванні нейронних мереж є Google Cloud Platform, AWS, Nvidia GPU Cloud.

Нейрокомп'ютери (комп'ютери на основі нейронних мереж) або нейроімітатори (програми, що моделюють роботу нейронних мереж не ЕОМ) мають цілу низку корисних властивостей: висока швидкодія за рахунок використання масового паралелізму обробки інформації; толерантність до

помилок – працездатність зберігається при пошкодженні значної кількості нейронів; здатність до навчання – програмування системи замінюється навчанням; здатність до розпізнавання образів в умовах сильних перешкод та спотворень. Однак перші 2 властивості мають місце тільки при апаратній реалізації нейронних мереж. Апаратно реалізовані нейронні мережі забезпечують вирішення складних завдань за часи порядку часів спрацьовування ланцюжків електронних та/або оптичних елементів.

Зараз нейронні мережі застосовуються для вирішення багатьох неформалізованих або важко формалізованих завдань:

- розпізнавання та синтезу мови;
- розпізнавання аерокосмічних зображень;
- прогнозування котирування цінних паперів та курсу валют;
- попередження шахрайства з кредитними картками;
- оцінки вартості нерухомості;
- оцінки фінансового стану підприємств та ризику неповернення кредитів;
- обробки радіолокаційних сигналів;
- контролю руху на швидкісних автомагістралях та залізницях;
- діагностики у медицині;
-видобутку знань із великих обсягів даних у бізнесі, фінансах та наукових дослідженнях.

Нейронні мережі можна використовувати за таких умов:

- якщо завдання може вирішувати людина;
- якщо при вирішенні завдання можна виділити множину вхідних факторів (сигналів, ознак, даних тощо) та множину вихідних факторів;
- якщо зміни вхідних факторів призводять до зміни вихідних.

Застосування нейронних мереж для вирішення окремих завдань може бути ефективнішим за використання людини. Це обумовлене тим, що людський розум орієнтований на вирішення завдань у тривимірному просторі. Проте багатовимірні задачі поставлені людині потребують значно більшої

трудомісткості. Штучним нейронним мережам не властиві подібні обмеження. Їм все одно вирішувати тривимірне або 10-мірне завдання.

Можна виділити основні тенденції розвитку машинного навчання в найближчому майбутньому:

- прориви в підходах до навчання без учителя;
- складніші архітектурні рішення, що базуються на різних взаємозамінних модулях;
- глибші моделі, що навчаються на меншій кількості прикладів;
- вирішення більш складних задач, таких як розуміння відео та обробка природньої мови

Для застосування нейронних мереж необхідно вирішити такі завдання:

-постановка задачі, придатної для розв’язання за допомогою нейронної мережі;

- вибір моделі ШНМ;

- підготовка вихідних даних для навчання ШНМ;

- навчання ШНМ;

- власне розв'язання задачі за допомогою навченої ШНМ.

Крім того, іноді потрібен ще один етап – інтерпретація рішення, отриманого нейронною мережею.

Найбільш трудомісткими процесами при використанні нейронних мереж є підготовка вихідних даних для навчання та навчання нейронної мережі.

Розроблений програмний засіб можна буде використовувати для проектування нейронних мереж на різних операційних системах та апаратних платформах для пришвидшення проектування нейронних мереж.

Висновки по першому розділу

Хоча рішення отримане на основі нейронної мережі може виглядати і поводитись як звичайне програмне забезпечення, вони різні в принципі, оскільки більшість реалізацій на основі нейронних мереж “навчається”, а не програмується: мережа навчається виконувати завдання, а не програмується безпосередньо. Рішення на основі нейронних мереж з часом стають все більш досконалими і, безсумнівно, у майбутньому наші можливості розробки відповідних пристроїв зростуть за рахунок кращого розуміння їх основоположних принципів. Але вже сьогодні є чимало вражаючих розробок. База додатків нейронних мереж просто величезна: виявлення фальшивих кредитних карток, прогнозування змін фондової біржі, оптичне розпізнавання символів, спостереження технічним станом механізмів, профілактика і діагностика захворювань людини, автоматичне управління рухом автомобіля тощо. Подальші успіхи у створенні штучних нейронних мереж залежатимуть від подальшого розуміння принципів роботи людського мозку, але тут є і зворотний зв'язок: штучні нейронні мережі є одним із засобів, за допомогою яких удосконалюється наше представлення про процеси, що відбуваються в нервовій системі людини.

РОЗДІЛ 2 МАТЕМАТИЧНІ МОДЕЛІ І АЛГОРИТМИ

Пошуки алгоритмів, що дозволять автоматично використовувати накопичених досвід, тривають більше 100 років. З появою доступних за вартістю комп’ютерів стався різкий стрибок в цій галузі. Виникли такі нові сфери діяльності - нейроінформатика, нейрокібернетика та ін. Основою роботи самонавчальних нейропрограм є нейронна мережа, що є сукупністю нейронів - елементів, пов'язаних між собою певним чином. Нейрони і міжнейронні зв'язки задаються програмно на звичайному комп'ютері або можуть мати "матеріальну" основу - особливі мікросхеми, які застосовуються в спеціально створених нейрокомп'ютерах. Функціонування нейрона в нейрокомп'ютері або нейропрограмі віддалено нагадує роботу біологічного нейрона. Біонейрон - клітина, що має довгі відростки, пов'язані з іншими нейронами за допомогою синапсів, що змінюють електричний імпульс від одного нейрона до іншого. Відростки підрозділяються на дендрит що передає сигнал до цього нейрона, і аксони що передають сигнали від цього нейрона.

2.1. Математична модель нейрона

Основний елемент нейронної мережі - це формальний нейрон. Нейрон є одиницею обробки інформації в нейронній мережі. На рисунку 2.1 приведена модель нейрона, що лежить в основі штучних нейронних мереж.

У цій моделі нейрона можна виділити три основні елементи:

- синапси, кожен з яких характеризується своєю вагою або силою. Здійснюють зв'язок між нейронами, множать вхідний сигнал u_i на ваговий коефіциент синапса w_i, що характеризує силу синаптичного зв’язку;
- суматор, аналог тіла клітини нейрона. Виконує складання зовнішніх вхідних сигналів або сигналів, що поступають по синаптичним зв'язкам від інших нейронів. Визначає рівень збудження нейрона;
- функція активації, визначає остаточний вихідний рівень нейрона, з яким сигнал збудження (гальмування) поступає на синапси наступних нейронів.

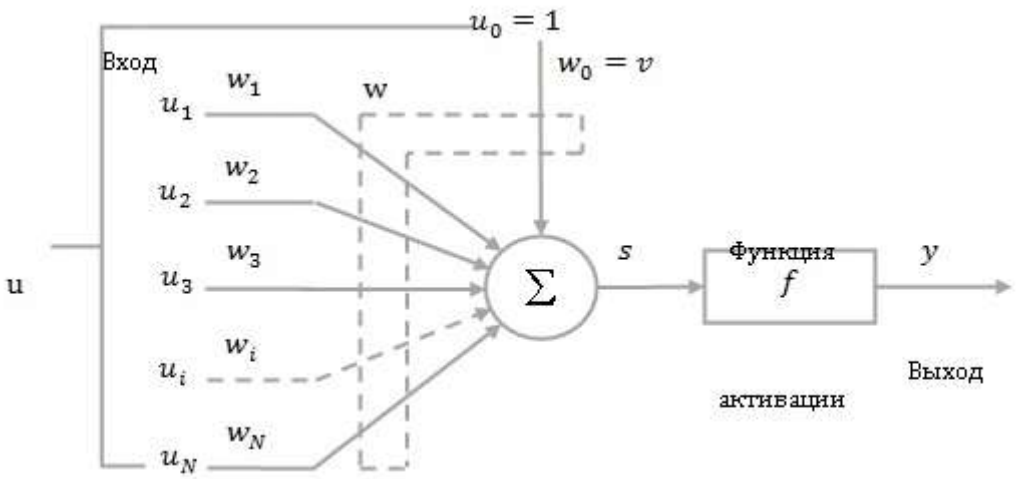

Рисунок 2.1 – Модель нейрона

Модель нейрона імітує в першому наближенні властивості біологічного нейрона. На вхід штучного нейрона поступає деяка множина сигналів, кожен з яких є виходом іншого нейрона. Кожен вхід множиться на його відповідну вагу, пропорційну синаптичній силі, і всі добутки підсумовуються, визначаючи рівень активації нейрона.

Хоч мережеві парадигми досить різноманітні, в основі їх більшості лежить саме ця модель нейрона. Тут множина вхідних сигналів, позначених $u_1, u_2, \ldots, u_N$ подається на штучний нейрон. Ці вхідні сигнали, що сукупно позначені вектором u, відповідають сигналам, що приходять в синапси біологічного нейрона. Кожен сигнал множиться на відповідну вагу $w_1, w_2, \ldots, w_N$ та передається на блок - суматор позначений Σ. Кожна вага відповідає «силі» одного біологічного синаптичного зв'язка. Множина ваги в сукупності позначається вектором w. Підсумовуючий блок, що відповідає тілу біологічного елемента, складає зважені входи алгебраїчно, створюючи вихід s. Далі s надходить на вхід функції активації f, і визначає остаточний сигнал збудження або гальмування нейрона на виході y. Сигнал збудження або гальмування визначається граничним значенням v. Цей сигнал надходить на синапс наступних нейронів і т.д.

Підіб'ємо підсумки всього вищесказаного:

• $u_1, \ldots u_N$ – вхідні сигнали даного нейрона, які надходять з інших нейронів;

• $w_1, \ldots w_N$ – синаптичні ваги;

• s – вихідне значення, отримане в результаті складання сигналів, що надходять із синаптичних зв'язків від інших нейронів;

• y – вихідний сигнал нейрона;

• v – граничне значення.

Вихідне значення s має вигляд

$$s = \sum_{i=0}^{N} w_i u_i$$

Формула,що описує функціонування нейрона, виглядає так

$$y = \begin{cases} 1 \text{ при } \sum_{i=1}^{N} w_i u_i \geq v \\ 0 \text{ при } \sum_{i=1}^{N} w_i u_i < v \end{cases}$$

або в іншій формі

$$y = f(\sum_{i=0}^{N} w_i u_i)$$

де

$$f(s) = \begin{cases} 1 \text{ при } s \geq 0 \\ 0 \text{ при } s < 0 \end{cases}$$

а також $w_0 = v\ u_0 = 1$

Ця проста модель нейрона не враховує багато властивостей свого біологічного двійника. Наприклад, вона не бере до уваги затримки в часі, що впливають на динаміку системи. Вхідні сигнали відразу породжують вихідний сигнал. Також дана модель нейрона не враховує впливів функції частотної модуляції або функції сінхронізації біологічного нейрона, які ряд дослідників вважають вирішальними. Незважаючи на ці обмеження, мережі, що побудовані на основі цієї моделі нейрона, виявляють властивості, які сильно нагадують біологічну систему. Тільки час і подальші дослідження зможуть відповісти на

запитання, чи подібні збіги є випадковими чи є наслідком того, що саме в цій моделі нейрона вірно схоплені найважливіші риси біологічного прототипу.

Нейрон здійснює операцію нелінійного перетворення суми добутку вхідних сигналів на вагові коефіцієнти:

$$y = F(\sum_{i=1}^{n} w_i x_i) = F(WX)$$

,

де X - вектор вхідного сигнала;

W – вектор ваги;

F - оператор нелінійного перетворення.

Сума добутків вхідних сигналів на вагові коефіцієнти називається зваженою сумою. Вона є скалярним добутком вектора ваг на вхідний вектор:

$$\mathrm{S} = \sum_{\mathrm{i}=1}^{\mathrm{n}} w_i x_i = (W, X) = |W||X|\cos\alpha$$

,

де |W|, |X| - відповідно довжини векторів W та X;

$\alpha = W, X$ - кут між векторами W й X.

Довжину вагового і вхідного векторів вмизначимо через їх координати:

$$|\mathrm{W}| = \sqrt{w_1^2 + w_2^2 + w_1^2 + ... + w_n^2}$$

;

$$|\mathrm{X}| = \sqrt{x_1^2 + x_2^2 + x_1^2 + ... + x_n^2}$$

.

Оскільки для нейронного елемента довжина вагового вектора після навчання | W | = const, то величина виваженої суми визначається проекцією вхідного вектора на ваговий вектор:

$$\mathrm{S} = |W||X|\cos\alpha = |W|X_w$$

,

Якщо вхідні вектори внормовані, тобто |X|=const, то величина зваженої суми залежатиме лише від кута між векторами X та W. Тоді при різних вхідних сигналах зважена сума буде змінюватися за косинусоїдальним законом. Максимального значення вона досягатиме при колінеарності вхідного та вагового векторів.

Якщо сила зв'язку w_i негативна, то такий зв'язок називається гальмуючим. В

іншому випадку синоптичний зв'язок є посилюючим.

Оператор нелінійного перетворення називається функцією активації нейронного елемента. В якості цього оператора можуть використовуватися різні функції, які вибираються відповідно до розв'язуваного завдання та типу нейронної мережі. До найпоширеніших функцій активації нейронних елементів відносять:

- Лінійна. Функція активності для вхідних елементів може бути тотожною функцією, і в цьому випадку вихідне значення нейронного елемента дорівнює:

$$y = kS,$$

де k - коефіцієнт нахилу прямої.

Вхідні елементи зазвичай призначені для розподілу сигналів, що вводяться між іншими елементами мережі, тому для вхідних елементів зазвичай потрібно, щоб вихідний від елемента сигнал був таким же, як і вхідний, тобто. k=1. Вхідні елементи мають лише одне вхідне значення. Наприклад, кожен елемент може отримувати сигнал від одного відповідного датчика, розміщеного десь на виробничому процесі. Один цей елемент зв'язується з багатьма іншими елементами мережі, тому дані, отримані від одного датчика, виявляються розподіленими між багатьма елементами мережі. Через те що вхідні елементи призначені виключно для того, щоб розподіляти сигнали, що отримуються із зовнішнього середовища, багато дослідників взагалі не вважають вхідні елементи частиною нейронної мережі.

- Порогова функція. Пороговою функцією активації може бути біполярна або бінарна функції. У разі використання порогової бінарної функції маємо:

$$y = \begin{cases} 1, S > \theta \\ 0, S < \theta \end{cases}$$

А якщо застосовується гранична біполярна функція активації, то вихідне значення нейронного елемента буде таким:

$$y = \begin{cases} 1, S > \theta \\ -1, S < \theta \end{cases}.$$

- Сігмоїдна функція. Ця функція є безперервною, зростаючою функцією в

діапазоні [0, 1]:

$$y = \frac{1}{1 + e^{-cS}},$$

де c>0 - коефіцієнт, що характеризує ширину функції по осі абсцис

Сігмоїдна функція є монотонною та всюди диференційованою, тому вона широко поширена в штучних нейронних мережах. Також є біполярна сігмоїдна функція, значення якої є в діапазоні [-1,1].

$$y = \frac{2}{1 + e^{-cS}} - 1,$$

Застосування різних функцій активації визначається класом задач, що розв'язуються нейронною мережею. Крім перелічених можуть застосовуватися інші функції активації нейронного елемента, які адекватно передають вирішуване завдання.

2.2. Види нейронних мереж

Нейронні мережі розрізняють структурою мережі (зв'язків між нейронами), особливостям моделі нейрона, особливостям навчання мережі.

За структурою нейронні мережі можна поділити на неповнозв'язні (або шаруваті) та повнозв'язкові.

Багатошарові (шаруваті) мережі: нейрони розташовані в кілька шарів, рисунок 2.2.

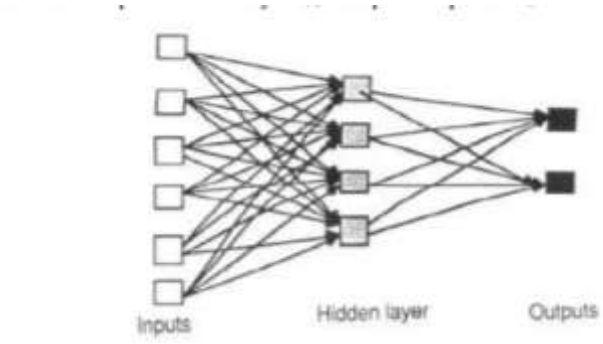

Рисунок 2.2. Багатошарова мережа

Нейрони з першого шару отримують вхідні сигнали, перетворюють їх та через точки розгалуження передають нейронам другого шару. Далі спрацьовує другий шар і т.д. до k-го, що видає вихідні сигнали для інтерпретатора та

користувача. Якщо не обумовлено інакше, то кожен вихідний сигнал і-го шару подається на вхід всіх нейронів і+1-го шару. Число нейронів у кожному шарі може бути будь-яким і ніяк не пов'язане з кількістю нейронів в решті шарів. Стандартний спосіб подачі вхідних сигналів: всі нейрони першого шару одержують кожен вхідний сигнал. Особливого поширення набули тришарові мережі, у яких кожен шар має свою назву: перший - вхідний, другий - прихований, третій - вихідний.

Повнозв'язні мережі: кожен нейрон передає свій вихідний сигнал іншим нейронам, у тому числі й самому собі

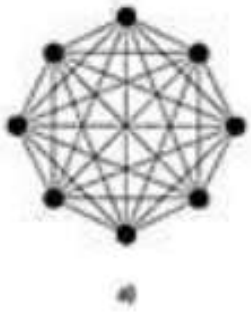

Рисунок 2.3 - Повнозвязна мережа

Вихідними сигналами для мережі можуть бути всі або деякі вихідні сигнали нейронів після кількох циклів функціонування мережі. Усі вхідні сигнали подаються всім нейронам. За сигналами, що використовуються на входах і виходах, нейронні мережі можна розділити на аналогові та бінарні.

За моделюванням часу нейронні мережі поділяються на мережі з безперервним і дискретним часом. Для програмної реалізації застосовується зазвичай дискретний час.

За способом подання інформації на входи нейронної мережі розрізняють:

1. подачу сигналів на синапси вхідних нейронів;
2. подання сигналів на виходи вхідних нейронів;
3. подачу сигналів у вигляді ваг синапсів вхідних нейронів;
4. адитивну подачу на синапси вхідних нейронів.

За способом знімання інформації з виходів нейронної мережі розрізняють:

1. знімання з виходів вихідних нейронів;

2. знімання з синапсів вихідних нейронів;
3. знімання у вигляді значень ваг синапсів вихідних нейронів;
4. адитивний знімання з синапсів вихідних нейронів

Залежно від функції активації нейронів мережа може бути однорідною та неоднорідною. Однорідна, або гомогенна, нейронна мережа це коли нейрони однаково перетворюють вихідний сигнал адаптивного суматора. Якщо ж функція активації залежить від одного або кількох параметрів, значення яких змінюються від нейрона до нейрона, то мережу називають неоднорідною або гетерогенною.

По організації навчання поділяють навчання нейронних мереж з учителем (supervised neural networks) і без вчителя (nonsupervised). При навчанні з учителем передбачено що існує зовнішнє середовище, яке надає навчальні приклади (значення входів та відповідні до них значення виходів). На етапі навчання вчитель оцінює правильність функціонування нейронної мережі, і відповідно до своїх критеріїв запускає механізм зміни стану нейронної мережі.

Під станом нейронної мережі, що може змінюватися, розглядають:

- ваги синапсів нейронів;

- встановлення нових зв'язків між нейронами (властивість біологічних нейронів встановлювати нові зв'язки та ліквідувати старі називається пластичністю).

За способом пред'явлення прикладів розрізняють пред'явлення одиночних прикладів та "сторінки" прикладів. У першому випадку зміна стану нейронної мережі (навчання) відбувається після пред'явлення кожного прикладу. У другому - після пред'явлення "сторінки" (набору) прикладів на основі аналізу відразу їх усіх.

2.3. Математична модель нейронної мережі

2.3.1. Одношарові нейронні мережі

Розглянемо нейронні мережі, що складені з одного шару нейронних елементів, який здійснює обробку вхідної інформації. Такі мережі прийнято зображати як двошарові нейронні мережі, де перший шар нейронних елементів є розподільним, а другий - опрацьовуючим. Розподільний шар передає вхідні

сигнали на опрацьовуючий шар нейронних елементів, який в свою чергу перетворює вхідну інформацію відповідно до синоптичних зв'язків і функції активації. При цьому кожен нейрон розподільчого шару має синоптичні зв'язки з усіма нейронами шару, що обробляє. Тоді вихідне значення для j-го нейронного елемента другого шару можна описати так:

$$y_j = F(S_j) = F(\sum_{i=1}^{n} w_{ij} x_i - T_j),$$

де T_j - поріг j-го нейронного елемента;

w_{ij} - сила синаптичного звязку між i-м нейроном розподільчого шару та j-м нейроном опрацьовуючого шару

Поріг нейронного елемента характеризує розташування функції активації по осі абсцис. При операціях з нейронними мережами поріг нейронного елемента вихідного шару, а також і прихованих шарів, можна виносити за межі і зображати як синоптичний зв'язок з ваговим коефіцієнтом, що дорівнює Т, і вхідним значенням, рівним -1.

Сукупність вагових коефіцієнтів мережі можна подати матрицею:

$$W = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1m} \\ w_{21} & w_{22} & \dots & w_{2m} \\ \dots & \dots & \dots & \dots \\ w_{n1} & w_{n2} & \dots & w_{nm} \end{bmatrix}.$$

Тоді вектор-стовпець зваженої суми в матричному вигляді буде:

$$S = W^T X - T,$$

де Т - вектор-стовпець порогів нейронних елементів другого слоя.

Розглянем як приклад нейронну мережу з двума нейронами вхідного та одним нейроном вихідного шару, рисунок 2.4.

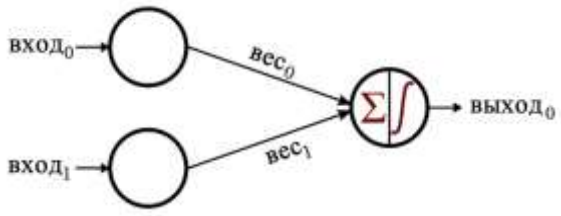

Рисунок 2.4. Одношарова мережа

Функція активації нейронів вихідного шару - порогова . Зважена сума:

$$S = w_{11}x_1 + w_{21}x_2 - T_1.$$

Відповідно значення на виході мережі:

$$y = \begin{cases} 1, S > 0 \\ 0, S < 0 \end{cases}.$$

Така мережа здійснює лінійний поділ вхідного простору на два класи, і може використовуватися для вирішення задач класифікації образів. При цьому рівняння лінії поділу:

$$w_{11}x_1 + w_{21}x_2 - T_1 = 0.$$

Вона відокремлює область рішень, що відповідає одному класу, від іншого класу і називається дискримінантною лінією. Маємо:

$$x_2 = \frac{T_1}{w_{21}} - \frac{w_{11}}{w_{21}} x_1.$$

У системі координат (x_2, x_1) це рівняння відповідає прямій лінії, яка відокремлює один клас від іншого. Тут величина T1/|W| є відстань від центру координат до прямої. Якщо розмірність вхідного сигналу n=3, то поверхнею, що розділяє, буде площина, якщо n>3 - гіперплощина.

2.3.2 Багатошарові нейронні мережі

Багатошарова нейронна мережа здатна здійснювати будь-яке відображення вхідних векторів у вихідні. Найбільшого поширення набули нейронні мережі з шаруватою архітектурою та сигмоїдною функцією активації нейронних елементів у всіх шарах крім вхідного. Функція активації нейронів вхідного шару, як зазначалося раніше, є лінійною.

Нехай W(i) - матриця вагових коефіцієнтів і-го шару багатошарової мережі. Тоді для нейронної мережі з двома прихованими шарами вихідні значення такі:

$$Y = F(F(F(X^{(1)}W^{(1)})W^{(2)})W^{(3)}),$$

де $X=(x_1, x_2,\ldots,x_n)$ - вектор-рядок вхідних сигналів;

F - оператор нелінійного перетворення.

Загальне число синоптичних звязків:

$$V = \sum_{i=1}^{p-1} k(i)k(i+1)$$
,

де p – загальна кількість шарів нейронної мережі;

k(i) - кількість нейронних елементів в і-м шарі.

Число шарів у багатошаровій нейронній мережі характеризує те, яким чином вхідний простір може бути розбитий на підпростор меншої розмірності. Так, двошарова нейронна мережа з одним шаром нелінійних нейронів розбиває вхідний простір образів на класи за допомогою гіперплощини. Тришарова нейронна мережа, в якій як два останніх шарів використовуються нейронні елементи з нелінійною функцією активації, дозволяє формувати будь-які опуклі області в просторі рішень. Чотиришарова нейронна мережа, яка має три нелінійні шари, дає можливість отримувати область рішень будь-якої форми та складності.

2.4. Навчання нейронної мережі

Навчання - це процес розвитку за допомогою взаємодії із зовнішнім середовищем та з урахуванням індивідуальності об’єкта. Самоадаптація та самоорганізація нейронних мереж досягається у процесі їх навчання, під час якого закріплюються синаптичні зв'язки між нейронними елементами. Навчальні правила визначають, як змінюються вагові коефіцієнти у відповідь на вхідний вплив.

2.4.1 Правило навчання Хебба

Правило навчання Хебба має біологічні засади. Воно є основою багатьох методів навчання нейронних мереж. Згідно з цим правилом, навчання відбувається внаслідок посилення сили зв'язку (синаптичної ваги) між одночасно активними нейронами. Виходячи з цього, зв’язки що частіше використовуються в мережі посилюються, це пояснює методику навчання шляхом повторення та звикання.

Нехай є два нейронні елементи і та j, між якими існує сила зв'язку, що дорівнює w_{ij}, тоді правило навчання Хебба запишеться так:

$$w_{ij}(t+1) = w_{ij}(t) + x_i y_j,$$

де t - час;

x_i и y_j- відповідно вихідне значення і-го та j-го нейронів.

У початковий момент часу передбачається, що:

$$w_{ij}(t = 01) = 0, \forall i, j.$$

Правило Хебба для нейронної мережі з попереднього прикладу, можна представити як:

$$w_{11}(t+1) = w_{11}(t) + x_1 y_1,$$
$$w_{21}(t+1) = w_{21}(t) + x_2 y_1,$$
$$T(t+1) = T(t) - y_1.$$

Подібним чином правило Хебба записується для нейронної мережі більшої розмірності. Правило Хебба може використовуватися як під час навчання з учителем, так і без нього. Якщо в якості вихідних значень для нейронної мережі використовуються еталонні значення, це правило буде відповідати навчанню з учителем. При використанні реальних значень, що виходять при подачі на вхід мережі вхідних образів, правило Хебба відповідає навчанню без вчителя.

В останньому випадку вагові коефіцієнти нейронної мережі в початковий момент часу ініціалізуються випадковим чином. Навчання з використанням правила Хебба закінчується після подачі всіх наявних образів на нейронну мережу. Слід зазначити, що у загальному випадку правило Хебба не гарантує збіжності процедури навчання нейронної мережі.

2.4.2 Правило Розенблатта

Цю процедуру запропонував американський вчений Ф. Розенблат. для нейронної мережі, яку він назвав персептрон. Персептрон - це мережа, що складається з S, A та R нейронних елементів. Нейрони шару S називають сенсорними, вони призначені для формування вхідних сигналів в результаті зовнішніх впливів. Нейрони шару A називаються асоціативними та призначені для безпосередньої обробки вхідної інформації. Нейрони шару R називають ефекторними. Вони служать передачі сигналів збудження до відповідного об'єкту-

приймача, наприклад до м'язів.

Таким чином, персептрон Розенблатта містить один шар оброблюючих нейронних елементів, в яких використовуються нейронні елементи з пороговою функцією активації. Тому навчання персептрона відбувається шляхом налаштування вагових коефіцієнтів між шарами S та A.

Математичне формулювання правила навчання Розенблатта можна подати у такому вигляді:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha x_i t_j,$$

де t_j - эталоне значення j-го виходу нейронної мережі;

α - коефіцієнт, що показує швидкість навчання.

Розмір швидкості навчання характеризується такими показниками:

$$\alpha = const,$$

$$0 < \alpha \le 1.$$

Процедура навчання Розенблатта містить наступні кроки:

1 Вагові коефіцієнти W нейронної мережі ініціалізуються випадковим чином або встановлюються в нульовий стан.

2 На входи мережі по черзі подаються вхідні образи X з навчальної вибірки, які трансформуються у вихідні сигнали нейронних елементів Y.

Якщо реакція нейронної мережі y_j збігається з еталонним значенням t_j, тобто. якщо $y_j=t_j$, то ваговий коефіцієнт w_{ij} не змінюється.

Якщо вихідна реакція мережі y_j збігається з еталонною, тобто. якщо $y_j \neq t_j$, то відбувається модифікація вагових коефіцієнтів.

Кроки 2-4 повторюються доти, доки стане $y_j=t_j$, для всіх вхідних образів, або перестануть змінюватися вагові коефіцієнти.

Відповідно до теореми збіжності персептрона, алгоритм сходиться за скінченну кількість кроків, якщо існує рішення задачі.

Основні відмінності процедури навчання Розенблатта від правила навчання Хебба полягають у наступному:

- У правилі навчання Розенблатта для персептрона є швидкість навчання α;
- Персептрон не змінює вагових коефіцієнтів, якщо вихідні сигнали збігаються з

еталонними;

- Вхідні образи з навчальної вибірки в моделі персептрона подаються доти, доки не відбудеться навчання мережі.

- Персептрон навчається за скінченну кількість кроків, якщо існує розв'язання задачі.

2.4.3 Правило Видроу-Хоффа

Правило навчання Відроу-Хоффа ще відоме під назвою дельта-правило. Воно передбачає мінімізацію середньоквадратичної помилки нейронної мережі, яка для L вхідних образів визначається наступним чином:

$$E = \sum_{k=1}^{L} E(k) = \frac{1}{2} \sum_{k=1}^{L} (y_1^k - t^k)^2$$

де E(k) - средньоквадратична похибка мережі для k-го образа;

y_{1k} и t^k - відповідно вихідне й еталонне значення мережі для k-го образа.

Даний критерий характерний тим, що при малих похибках ушкодження є також малим, бо Е менше величини відхилення (y-t)..

Середньоквадратична помилка нейронної мережі для одного вхідного образу визначається за такою формулою:

$$E(k) = \frac{1}{2} (y_1^k - t^k)^2 .$$

Правило навчання Видроу-Хоффа виходить з методики градієнтного спуску у просторі вагових коефіцієнтів і порогів нейронної мережі. Відповідно до цього правила вагові коефіцієнти та пороги нейронної мережі необхідно змінювати з часом за такими виразами:

$$w_{j1}(t+1) = w_{j1}(t) - \alpha \frac{\partial E(k)}{\partial w_{j1}(t)};$$

$$T(t+1) = T(t) - \alpha \frac{\partial E(k)}{\partial T(t)},$$

де $j = \overline{1,n}$;

α – швидкість або крок навчання.

Б.Відроу та М.Хофф довели, що таке правило навчання завжди дозволяє знаходити вагові коефіцієнти нейронного елемента таким чином, щоб

мінімізувати середньоквадратичну помилку мережі незалежно від початкових значень цих вагових коефіцієнтів.

У цьому правилі є проблема вибору значення для кроку навчання α. Якщо цей крок α замалий, процес навчання буде дуже тривалим. Якщо α великий, то процес навчання може виявитися розбіжним, тобто. не призведе до вирішення задачі. Отже, збіжність навчання позбавляє розумного вибору значення кроку навчання.

2.4.4. Алгоритм зворотного розповсюдження помилки

Алгоритм зворотного поширення помилки є одним із найбільш ефективних засобів для навчання багатошарових нейронних мереж. Алгоритм мінімізує середньоквадратичну помилку мережі, і ґрунтується на наступних теоремах.

Теорема 1. Для будь-якого прихованого шару і помилка і-го нейронного елемента визначається рекурсивним чином через помилки нейронів наступного шару j:

$$\gamma_i = \sum_{j=1}^{m} \gamma_j F'(S_j) w_{ij},$$

де m - число нейронів наступного шару відносно до і;

w_{ij} - синаптичний звязок між і-м та j-м нейронами різних шарів;

S_j - зважена сума j-го нейрона.

За результатами цієї теореми, можна визначити помилки нейронів прихованого шару через помилки нейронів наступного шару відносно прихованого шару.

Теорема 2. Похідні середньоквадратичної помилки за ваговими коефіцієнтами та порогами нейронних елементів для будь-яких двох шарів і та j багатошарової мережі визначаються наступним чином:

$$\frac{\partial E}{\partial w_{ij}} = -\gamma_j F'(S_j) y_i;$$

$$\frac{\partial E}{\partial T_j} = \gamma_j F'(S_j).$$

На основі цих математичних доведень створено алгоритм зворотного

поширення похибки. В алгоритмі прийнято такі позначення, що відповідні і підпрограмі, що реалізує цей алгоритм:

- α – крок навчання;
- Em – необхідна середньоквадратична помилка;
- ероха - ціле число, що дорівнює кількості навчальних зразків;
- w_{ij} - ваговий коефіцієнт зв'язку між і-м та j-м нейронами суміжних шарів;
- st_Ar_IN – масив вхідних значень навчального зразка;
- st_Ar_OUT – масив вихідних значень навчального зразка;
- st_Ar - об'єкт класу StadyArrays, що зберігає st_Ar_IN та st_Ar_OUT разом;
- masyv_Study - масив об'єктів, що містять об'єкти класу StadyArrays;
- END - логічна змінна, що відповідає закінченню навчання;
- layers - ціле число, що дорівнює кількості всіх шарів у нейронній мережі;
- kilkist - ціле число, що дорівнює кількості нейронів у шарі;
- y - значення результату роботи нейронного елемента;
- rez_Stud - масив результатів всіх нейронів шару;
- result – масив цільових значень;
- err- помилка результату роботи нейрона;
- e[i] - сума квадратів помилок вихідного шару для і-го зразка;
- E – середньоквадратична помилка мережі для всіх зразків.

Дії алгоритму:

1. Користувач вводить бажані значення кроку навчання (α), середньоквадратичної помилки (Em) та кількість зразків для навчання (ероха).

2. Ініціалізуються випадковим чином вагові коефіцієнти (w_{ij}) нейронних елементів у межах від -0,5 до +0,5.

3. Початок циклу введення навчальних зразків.

4. Користувач вводить масив даних, що подається на вхід нейронної мережі, st_Ar_IN.

5. Користувач вводить масив цільових вихідних значень st_Ar_OUT.

6. З масивів st_Ar_IN і st_Ar_OUT створюється об'єкт класу StadyArrays.

7. Створений об'єкт додається до масиву masyv_Study.

8. Значення END встановлюється у “false”, мається на увазі що мережа неспроможна правильно вирішувати поставлене завдання, тобто. не навчена.

9. Перевірка параметра зупинки процесу навчання. Якщо END=”true”, цикл припиняється (перехід до кроку 30), інакше перехід до кроку 11.

10. Початок циклу навчання всім зразкам епохи.

11. Зчитується вхідний вектор значень чергового зразка. Задається res_Study.

12. Цикл прямого проходження інформації з нейронної мережі. Відбувається послідовне проходження інформації, що вводиться по всіх шарах.

13. Цикла обчислення реакції кожного нейрона в шарі на інформацію, що подається на цей шар.

14. Обчислюється результат роботи кожного нейрона:

$$y_j = F(\sum_i w_{ij} y_i),$$

де i - індекс, що характеризує нейрони попереднього шару відносно шару j. Для вхідного шару y_i береться із заданого в блоці 12 масиву результатів rez_Study.

15. Перевизначається масив rez_Study, для представлення його як вхідної інформації наступного шару.

16. Зчитується цільовий вектор значень поточного зразка. Задається результат.

17. Цикл обчислення помилок всіх нейронних елементів вихідного шару.

18. На підставі отриманого масиву result і обчисленого res_Study знаходяться помилки:

$$\gamma_j = y_j - t_j;$$

де y_j – обчислене значення, тобто. елемент масиву rez_Study;

t_j - відоме цільове значення елемент масиву result.

19 Цикл обчислення помилок у всіх прихованих шарах, починаючи з останнього.

20 Цикл обчислення помилок всіх нейронних елементів поточного прихованого шару.

На підставі отриманого обчисленого масиву наступного шару res_Study

знаходяться помилки:

$$\gamma_j = \sum_{i=1}^{m} \gamma_i F'(S_i) w_{ij};$$

де індекс і характеризує нейронні елементи наступного шару;

22 Цикл зміни вагових коефіцієнтів між усіма суміжними шарами.

23 Цикл для поточного шару за всіма його нейронними елементами.

Зміна вагових коефіцієнтів, знаходиться як:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha \gamma_j F'(S_j) y_i;$$

25 Обчислюється сума квадратів похибок для даного зразка навчання:

$$e = \sum_j (y_j - t_j)^2$$

Далі йде перехід до пункту 12 для зчитування наступного зразка в епосі навчання. Якщо це був останній зразок, виконується блок 27.

26 Знаходиться середньоквадратична помилка нейронної мережі за всіма зразками:

$$E = \frac{1}{2} \sum_{k=1}^{L} \sum_j (y_j^k - t_j^k)^2 = \frac{1}{2} \sum_i e_i,$$

де L – розмір вибірки для навчання.

27 Перевірка на досягнення середньоквадратичною помилкою допустимого значення. Безперечний подальший перехід до блоку 10.

28 При істинності блоку 28 значення END так само встановлюється “true”. Безперечний подальший перехід до блоку 10.

Недоліки алгоритму зворотного розповсюдження помилки. Оскільки даний алгоритм ґрунтується на методі градієнтного спуску в просторі вагових коефіцієнтів і порогів нейронної мережі, існує ряд проблем при навчанні. До таких проблем можна віднести такі:

- невизначеність з вибором числа шарів та кількості нейронних елементів у шарі для багатошарових мереж;
- повільна збіжність градієнтного методу із постійним кроком навчання;
- складність вибору відповідної швидкості навчання α. Так, занадто мала

швидкість навчання збільшує час навчання і призводить до скочування нейронної мережі в локальний мінімум. Велика швидкість навчання може призвести до пропуску глобального мінімуму і зробити процес навчання таким, що розходиться;

- неможливість визначення точок локального та глобального мінімумів, оскільки градієнтний метод їх не розрізняє;
- вплив випадкової ініціалізації вагових коефіцієнтів нейронної мережі на пошук мінімуму функції середньоквадратичної помилки.

Останній пункт показує, що з різної ініціалізації синоптичних зв'язків можуть виходити різні рішення завдання. Те, що алгоритм зворотного розповсюдження помилки не дозволяє в загальному випадку досягти глобального мінімуму, не применшує його переваг, оскільки в багатьох практичних завданнях достатньо навчити нейронну мережу до середньоквадратичної помилки. Чи є при цьому знайдений мінімум локальним чи глобальним, не принципово.

2.5. Рекомендации по вибору правил навчання і архітектури нейронних мереж.

Ефективність навчання багатошарових нейронних мереж залежить від кількості шарів, числа елементів у прихованих шарах нейронної мережі та початкової ініціалізації вагових коефіцієнтів.

Як зазначено вище, різна ініціалізація вагових коефіцієнтів може призводити до різних рішень. Важливу роль тут грає розмір синаптичних зв'язків, що ініціалізуються випадково. Це призводить до того, що процес навчання зупиниться у найближчому локальному мінімумі від стартової точки. Рекомендується випадковим чином ініціалізувати вагові коефіцієнти та порогові значення в діапазоні [-0,5; 0,5].

Велику роль в ефективності навчання відіграє і архітектура нейронної мережі. Розмірність вхідного та вихідного шарів нейронної мережі визначається з умови розв'язуваної задачі або навчальної вибірки. За допомогою тришарової нейронної мережі можна апроксимувати будь-яку функцію зі скільки завгодно

заданою точністю. При цьому точність апроксимації залежить від числа нейронів у прихованому шарі.

Чим більше число нейронних елементів у прихованому шарі, тим більша точність. Однак при занадто великій розмірності прихованого шару може настати явище, що називається перетренуванням мережі. Це означає, що мережа добре апроксимує функцію тільки на тренувальних зразках, але погіршується узагальнююча здатність мережі. З іншого боку, при надто малій розмірності прихованого шару можна потрапити в небажаний локальний мінімум або процес навчання буде надто тривалим. Тому тут потрібен розумний компроміс.

Для забезпечення необхідної точності та узагальнюючої здатності можна використовувати нейронну мережу з двома прихованими шарами, розмірність яких менша, ніж при використанні тришарової мережі. Але нейронні мережі, які мають кілька прихованих шарів, навчаються значно повільніше. Також одним із недоліків методу градієнтного спуску є застрягання в локальних мінімумах.

Висновки до другого розділу

Виходячи з наведених в розділі міркувань можна зробити наступні висновки:

- нейронна мережа з одним прихованим шаром дозволяє здійснити будь-яке відображення вхідних сигналів у вихідні;
- число нейронних елементів у проміжному шарі має бути меншим від числа тренувальних зразків;
- потужність нейронної мережі можна збільшувати як за рахунок числа нейронів у шарі, так і за рахунок числа шарів. Якщо на нейронну мережу накладається обмеження N2 і вона не може вирішити поставлену задачу, необхідно збільшувати кількість прихованих шарів;
- випадкова ініціалізація вагових коефіцієнтів нейронної мережі має відбуватися у доволі вузькому діапазоні значень.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Вибір програмних засобів.

Для програмної реалізації було обрано мову програмування Java як представника об'єктно-орієнтованих мов.

Об'єктно-орієнтоване програмування нині стало домінуючою парадигмою програмування, витіснивши “структурні”, процедурно-орієнтовані методи програмування. Мова Java є повністю об'єктно-орієнтованою мовою, і нею неможливо програмувати в процедурному стилі.

Як побачимо далі, весь код, що створюється в мові Java, знаходиться всередині класів. Стандартна бібліотека мови Java містить кілька тисяч класів, призначених для вирішення різних завдань, наприклад, для створення інтерфейсу користувача, календарів і т.д. Незважаючи на це, програмісти продовжують створювати свої власні класи на мові Java, щоб описати об'єкти, характерні для програми, що розробляється, а також адаптувати класи зі стандартної бібліотеки для своїх потреб.

Інкапсуляція – це ключове поняття під час роботи з об'єктами. Формально інкапсуляція - це просто об'єднання даних та операцій над ними в одному пакеті та приховування даних від користувача об'єкта. Дані в об'єкті називаються полями екземпляра, а функції та процедури, що виконують операції над даними, – його методами. У цьому об'єкті, тобто. екземплярі класу, поля екземпляра мають певні значення. Безліч цих значень називається поточним станом об'єкта. Застосування будь-якого методу якогось об'єкта може змінити його стан.

У процедурно-орієнтованому програмуванні спочатку формується завдання, та був з допомогою покрокового уточнення вихідне завдання розбивається все більш дрібні подзадачи, доки вони стануть настільки простими, що можна буде реалізувати безпосередньо. Потім ці вже реалізовані процедури об'єднуються у складніші конструкції, поки програма стане робити саме те, що потрібно (програмування “знизу вгору”).

При объектно-ориентированном програмуванні (ООП) у проекті спочатку виділяються класи, і потім визначаються їх методи. Для вирішення невеликих

завдань розбиття програми на процедури є цілком виправданим. Однак при розробці великих проектів класи та методи мають дві переваги. Класи надають зручний механізм кластеризації методів. Дуже простий Web-браузер для реалізації може вимагати 2000 функцій і 100 класів, тобто. у середньому 20 методів у класі. Програмістові набагато легше зрозуміти саме останню структуру. Крім того, її легше розподілити у команді програмістів. Цьому сприяє також інкапсуляція: класи приховують деталі представлення даних будь-якого коду, крім своїх методів.

У той час як на основі класу можна створити кілька об'єктів з однаковою поведінкою, у процедурно-орієнтованих мовах неможливо отримати кілька копій одного модуля. Не можна просто двічі викликати модуль. Для цього потрібно скопіювати та перейменувати всі процедури! Класи немає таких обмежень. Визначивши одного разу клас, легко створити будь-яку кількість його екземплярів (у той час як модуль має лише один екземпляр).

Ще один важливий принцип, що забезпечує високу продуктивність ООП, полягає в тому, що якщо перед одним об'єктом стоїть завдання, яке він не вирішує, у нього має бути доступ до іншого об'єкта, який зможе це вирішити. Перший об'єкт просить другого вирішити це завдання. Це узагальнений варіант виклику функцій, що застосовується в процедурному програмуванні. У мові Java зазвичай називається викликом методу. Об'єкт ніколи не повинен безпосередньо маніпулювати внутрішніми даними іншого об'єкта, а також надавати іншим об'єктам прямий доступ до своїх даних. Всі зв'язки між ними забезпечуються за допомогою методів викликів. Інкапсуляція даних об'єкта максимально підвищує можливість його повторного використання, зменшує їхню взаємозалежність та мінімізує час налагодження програми.

Ще одна з переваг мови Java - незалежність від платформи, на якій виконується програма: той самий код можна запустити під керуванням операційних систем Windows, Solaris, Linux, Macintosh та ін.

Інша перевага полягає в тому, що синтаксис мови Java схожий на синтаксис мови C++, і програмістам, знайомим з мовами C і C++, його вивчення не важко.

Крім того, Java - повністю об'єктно-орієнтована мова, навіть більшою мірою, ніж C++. Усі сутності у мові Java є об'єктами, крім небагатьох основних типів, наприклад чисел.

Розробники мови Java довго розмірковували про те, чому програми, написані на мову C++, так схильні до помилок. Вони забезпечили мову Java засобами, що дозволяють виключити саму можливість створювати програми, в яких були б приховані найпоширеніші помилки. Для цього в Java зроблено таке.

- Виключена можливість явного виділення та звільнення пам'яті. Пам'ять у мові Java звільняється автоматично за допомогою механізму збирання сміття. Програміст гарантований від помилок, пов'язаних із неправильним використанням пам'яті.
- Введені справжні масиви та заборонена арифметика покажчиків. Тепер програміст у принципі неспроможна стерти дані з пам'яті внаслідок неправильного використання покажчиків.
- Виключена можливість переплутати оператори присвоєння з оператором порівняння на рівність. Тепер не можна навіть скомпілювати вираз if (ntries = 3) ... (програмісти мовою Visual Basic можуть взагалі не помітити тут жодної проблеми, оскільки ця помилка - джерело більшості непорозумінь у мовах C і C++).
- Виключено множинне спадкування. Воно замінено новим поняттям - інтерфейсом, запозиченим з мови Objectiv C. Інтерфейс дає програмісту майже усе, що може отримати від множинного успадкування, уникаючи у своїй складнощів, що виникають під час управління ієрархіями класів.

Автори мови Java написали керівництво, в якому пояснювалися цілі його розробки та гідності мови. У цьому документі наведено одинадцять характерних рис Java. Розглянемо деякі з них.

Простота. Java - це система, за допомогою якої легко програмується, причому не потрібно додаткового навчання і враховується практика і стандарти програмування, що склалася. Тому мова Java був розроблений максимально схожим на мову C++, незважаючи на те, що самі розробники вважали мову C++

непридатною для цих цілей. Інший аспект простоти – стисліcть. Одна з цілей мови Java – забезпечити розробку програм, які можна було б самостійно виконувати на невеликих машинах. Розмір основного інтерпретатора та засобів підтримки класів складає близько 40 Кбайт; стандартні бібліотеки та засоби підтримки потоків займають ще 175 Кбайт.

Надійність. Мова Java призначена для створення програм, які повинні надійно працювати в будь-яких ситуаціях. Основна увага в мові Java приділяється ранньому виявленню можливих помилок, динамічній перевірці, а також виключенню ситуацій, схильних до помилок.

Не залежить від архітектури. Компілятор генерує об'єктний файл, формат якого залежить від архітектури комп'ютера, - скомпільована програма може виконуватися будь-яких процесорах під управлінням системи виконання програм мови Java. Для цього компілятор Java генерує команди байт-коду. Байт-код розроблений таким чином, щоб на будь-якій машині його можна було легко інтерпретувати або на льоту перевести в машинно-залежний код.

Розподіленість. Мова Java має велику бібліотеку програм для передачі даних на основі таких протоколів як TCP/IP, HTTP або FTP. Програми, написані мовою Java, можуть відкривати об'єкти та отримувати доступ до них через мережу за допомогою URL-адрес так само легко, як і в локальній мережі.

Безпечність. Мова Java призначена для використання в мережевому або розподіленому середовищі. З цієї причини велика увага була приділена безпеці. Мова Java дозволяє створювати системи, захищені від вірусів та стороннього втручання.

Багатопоточність забезпечує кращу інтерактивність та виконання програм.

3.2 Розробка об'єктів, що складають нейронну мережу.

У тексті розділу наводяться лише фрагменти програмного коду, які дозволяють продемонструвати загальну логіку. Докладніше опис модулів програми наведено у додатку А.

3.2.1 Модуль нейрона

Формальний нейрон згідно з описом складається з адаптивного суматора та

перетворювача виваженої суми. Також у складі нейронного елемента може бути точка розгалуження. Крім того, кожен нейронний елемент характеризується кількістю входів та виходів. Конструктор класу нейрона повинен відбивати всі ці особливості. Коротко конструктор виглядає так:

```
public Neiron (int IN, int OUT, String type)
{= OUT;(type == "in")
{= 1;_koef = new float [vxodi];(int i = 0; i < vxodi; i++)
{_koef[i]=1.0F;
}
}
{= IN+1;_koef = new float [vxodi];(int i = 0; i < vxodi; i++)
{= Math.random();(vk<=0.45) { ves_koef[i]= (float)vk;}_koef[i]=(float)(0.45-vk);
}
}
}
```

Для створення об'єкта потрібно вказати такі параметри:

IN – зберігає кількість входів у даний нейрон;

OUT - зберігає кількість виходів із даного нейрона;

type – тип нейрона.

Нейронний елемент може бути вхідним, прихованим або вихідним, відповідно до якого шару нейронної мережі він належить.

У кожному нейронному об'єкті зберігається масив вагових коефіцієнтів всіх зв'язків, що входять до нього. Поріг нейронного елемента в цій програмі відповідає ваговому коефіцієнту зв'язку, від елемента з постійним значенням виходу рівного 1. Для всіх нейронів не вхідного шару кількість входів, а значить і кількість елементів у масиві вагових коефіцієнтів, дорівнює кількості нейронів у попередньому шарі, збільшеному на 1. Ці вагові коефіцієнти задаються випадковим чином, використовуючи метод Math.random(), у межах [-0,45; 0,45]. Саме на відміну вхідного нейрона від інших у конструкторі і передбачено третій параметр type.

Якщо параметр type вказує на те, що нейронний елемент є частиною вхідного шару, кількість його входів дорівнює 1, і ваговий коефіцієнт цього входу дорівнює 1.

Для функціонування нейронного елемента необхідно передбачити реалізацію суматора та перетворювача. Ці методи описані по-різному залежно від того, чи є нейронний елемент вхідним чи ні. Для вхідного нейрона методи виглядають так:

```
public double getINResult (double tempResult)
{= 0.0;= sum + ves_koef[0]*tempResult;= getINResultat(sum);result;
}
double getINResultat (double param)
{res = param;res;
}
```

Для элементов скрытых и выходного слоев:

```
double getResult (double [] tempResult)
{= 0.0;(int i=0; i < vxodi; i++)
{(i < (vxodi-1)) sum = sum + ves_koef[i]*(tempResult[i]);sum = sum + ves_koef[i]*1;
}= getResultat (sum);
result;
}
double getResultat (double param)
{res = 1.0/(1.0 + Math.exp(-param));
return res;
}
```

Метод getResult в обох випадках повертає зважену суму, причому параметром є результат функціонування попереднього шару, у разі вхідного шару – це дані, що вводяться користувачем.

У процесі навчання необхідно мати доступу до полів об'єкта, тобто. змінювати вагові коефіцієнти нейрона. Для цього передбачено метод такого доступу:

```
public void setVesKoef (int j, double tmp)
{temp = ves_koef [j] + tmp;_koef[j] = (float)temp;
```

```
}
```

Інші методи, наведені у додатку А.

3.2.2 Модуль для шарів нейронної мережі

У будь-якій нейронній мережі є вхідний шар та вихідний шар, і може бути кілька прихованих шарів. Тому, кожен шар створюється з параметром tip, який визначає обмеження на функціонування даного шару і всіх нейронних елементів шару. Розглянемо конструктор для об'єкта класу, що описує шар нейронної мережі. Крім параметра tip, тут треба вказати кількість нейронних елементів у даному шарі – capacity_Neironiv, та vxodi – кількість елементів у попередньому шарі. Цей параметр потрібен для правильного створення об'єктів нейронних елементів у поточному шарі, оскільки він відповідає кількості входів у кожен нейрон. Зауважимо, що для вхідного шару цей параметр не враховується, оскільки там не існує попереднього шару.

```
public Layer (String tip, int capacity_Neyroniv, int vxodi)
{estvo = capacity_Neyroniv;= new Array_List(kilkist);.toLow_Case();(tip == "in")
{= kilkist;= 1;(int i=0; i<kilkist; i++)
{neiron1 = new Neyron (1, output, "in");.add(neyron1);
}.trim_ToSize();
}
{(tip == "out")
{= vxodi;= kilkist;(int i=0; i< kilkist; i++)
{neyron1 = new Neyron (input , 1, "out");.add(neyron1);
}.trim_ToSize();
}
{(tip == "sub")
{= vxodi;= 1;(int i = 0; i < kilkist; i++)
{neyron1 = new Neyron (input, output, "sub");.add(neyron1);
}.trimToSize();
}
}
}
}
```

Кожен шар має такі поля як:

список масивів masiv_Neyron, що містить об'єкти нейронних елементів;

масив mas_Ras - масив результатів функціонування шару, тобто. значення функції активації кожного нейронного елемента шару;

масив mas_err - масив помилок всіх нейронних елементів шару у процесі навчання.

Список масивів masiv_Neiron заповнюється елементами під час створення об'єкта шару нейронної мережі, два останні масиви спочатку ініціалізуються та встановлюються рівними null. Масив mas_Ras використовується як для навчання, так і при подальшому функціонуванні нейронної мережі, а mas_err необхідний тільки на стадії навчання.

Для розрахунку результату використовується окремий метод. При виклику цього методу вказується порядковий номер шару моделі нейронної мережі, і значення, що надходять на цей шар. Порядковий номер служить для того, щоб відрізнити вхідний шар від інших. т.к. Способи нейронів вхідного шару різняться від методів інших нейронів.

```
public double[] makeRas4et (int thisSloi, double [] lastSloiResult)
{et = new double[koli4estvo];(int i =0; i<koli4estvo; i++)
{n = (Neiron) massivNeironov.get(i);(thisSloi==0)
{= lastSloiResult[i];= n.getINResult (temp);
}
{= n.getResult (lastSloiResult);
}et[i]=temp;
}
return massivRas4et;
}
```

Під час навчання, згідно з алгоритмом зворотного розповсюдження помилки, необхідно буде коригувати вагові коефіцієнти. Для цього є метод:

```
public void izmenenieVesov (double normaObu4, double [] resultPreL, double [] oshibki)
{(int i = 0; i < koli4estvo; i++)
{n = (Neiron) massivNeironov.get(i);(int j =0; j<=resultPreL.length; j++)
{(j!=resultPreL.length)
```

```
{= oshibki[i] * normaObu4 * resultPreL[j];.setVesKoef(j, temp);
}
{= oshibki[i] * normaObu4 * 1;.setVesKoef(j, temp);
}
}
}
}
```

Інші методи даного класу наведені в додатку і забезпечують реалізацію класу.

3.2.3 Створення класу нейронної мережі

Створення об'єкта класу нейронної мережі відбувається за допомогою схематичного зображення всіх шарів у вікні програми, з точним вказівкою кількості нейронних елементів у кожному шарі. Виділимо такі параметри:

- aIn - ціле число, що дорівнює кількості нейронів у вхідному шарі;
- aSubS - ціле число, що дорівнює кількості прихованих шарів;
- aSub - цілісний масив, елементами якого є значення кількості нейронних елементів у кожному із прихованих шарів;
- aOut - ціле число, що дорівнює числу нейронів у вихідному шарі;

```
Net (Frame owner, int aIn, int aSubS, int [] aSub, int aOut)
{= aIn;= aSubS;= new int [subS];(int i =0; i<subS; i++)
{[i] = aSub[i];
}= aOut;= 2 + subS;= new ArrayList ();(int i =0; i<layers; i++)
{(i == 0)
{= new Layer ("in", in, in);.add(l);
}
{(i != layers-1)
{preL = (Layer)massivLayers.get(i-1);v = sub[i-1];.setVIxodi(v);= new Layer ("sub", sub[i-1], preL.getVIxodi());.add(l);
}
{preL = (Layer)massivLayers.get(i-1);.setVIxodi(out);= new Layer ("out", out, preL.getVIxodi());.add(l);
}
}
```

}= new VvodKartinki (owner, in);

}

У конструкторі класу, присвоюються значення таким полям об'єкта як кількість нейронів у вхідному, вихідному та прихованих шарах, заповнюється список масивів масивів - це масив, в якому зберігаються всі об'єкти класу шару нейронної мережі, і створюється об'єкт діалогового вікна VvodKartinki для введення інформації, за якою нейронна мережа зможе або вчитися або робити розрахунок.

Для того щоб зберегти модель нейронної мережі у файл, передбачений метод, який збереже нейронну мережу як набір масивів: кількість шарів, кількість нейронів у кожному шарі, кожен шар як масив нейронів.

```
public void SaveNet (String fileToSave)
{fileName = fileToSave;.toLowerCase();(!fileName.endsWith(".net"))
{n = fileName.length();= fileName.substring(0,n) + ".net";
}[] sloi = new int [layers+1];[0] = in;[1] = subS;(int i = 0; i < subS; i++)
{[i+2] = sub[i];
}[2+subS] = out;
{outNet = new ObjectOutputStream (new FileOutputStream(fileName));.writeObject (sloi);(int i=0; i < layers; i++)
{= (Layer) massivLayers.get (i);.writeObject (l);
}.close ();
}(Exception e)
{.printStackTrace ();
}
}
```

Щоб завантажити нейронну мережу з файлу в класі нейронної мережі є особливий конструктор, який значення зчитує з файлу, шлях до якого заданий параметром.

public Net (Frame owner, String fileToOpen)

{

{inNet = new ObjectInputStream (new FileInputStream(fileToOpen));[] newSloi = (int[]) inNet.readObject();= newSloi[0];= newSloi[1];= new int [subS];(int i=0; i < subS; i++)

```
{[i] = newSloi[i+2];
}= newSloi[2+subS];= subS+2;= new ArrayList (layers);(int i=0; i<layers; i++)
{= (Layer) inNet.readObject();.add(l);
}.close();
}(Exception e)
{.printStackTrace();
}= new VvodKartinki (owner, in);
}
```

3.2.4 Модуль навчання з використанням алгоритму зворотного розповсюдження помилки

Реалізація відповідає алгоритму, наведеному у другому розділі роботи. Повністю метод наведено у додатку А. Частина алгоритму зворотного поширення помилки представлена нижче.

```
public void Study (JTextArea txtArea)
{.append ("\n\n Навчання мережІ\n"); = JOptionPane.showInputDialog ("Введіть величину припустимої середньоквадратичної похибки");
Em = Double.parseDouble (input);
input = JOptionPane.showInputDialog ("Введіть норму навчання в межах  0 до 1");
normaObu4eniya = Double.parseDouble (input);
input = JOptionPane.showInputDialog ("Скільки буде зразків в наборі?");
int epoxa = Integer.parseInt (input);
massivStudy = new ArrayList ();= (Layer) massivLayers.get (0);= l.getVIxodi();= (Layer) massivLayers.get (massivLayers.size()-1);= l.getVIxodi();(int j = 0; j < epoxa; j++)
{= new double [stIn];(vvod.showIt())
{= vvod.getMassiv();
}
{.append ("\n Виникла помилка\n");;
}= new double [stOut];.setMessage("Що маємо на виході мережі");
if(vvodCifr.showIt())
{= vvodCifr.getMassiv();
}
{.append ("\n Виникла помилка\n");;
}stAr = new StudyArrays (stIn, studyArrayIN, stOut, studyArrayOUT);.add(stAr);
```

```
}.trimToSize();(!STOP)
{= new double [epoxa];(int i =0; i < epoxa; i++)
{s = (StudyArrays)massivStudy.get(i);= new double [s.getIN()];= s.getArrayIn();(int j =0; j<layers; j++)
{= (Layer) massivLayers.get (j);(j==0)
{= new double[l.getVIxodi()];= l.makeRas4et (j, tempArray);
}
{= new double[l.getVIxodi()];= l.makeRas4et (j, resultStudy);= new double[l.getVIxodi()];(int ij = 0; ij<resultStudy.length; ij++)
{[ij] = tempArray[ij];
}
}
}= new double [s.getOUT()];= s.getArrayOut();(int j =layers-1; j>0; j--)
{= (Layer) massivLayers.get (j);(j == (layers-1))
{.ras4etOshibki (result, resultStudy);
}
{= (Layer) massivLayers.get (j+1);= new double [l.getVIxodi()];= l.getMassivRas4et();.ras4etOshibki (l1, resultStudy);
}
}
//обчислення квадрата похибки= new double [s.getOUT()];= s.getArrayOut();= (Layer) massivLayers.get (massivLayers.size()-1);= new double [l.getVIxodi()];= l.getMassivRas4et();last = true;= 0;(int j=0; j < l.getVIxodi(); j++)
{= temp + (tempArray2[j]- tempArray[j])*(tempArray2[j]- tempArray[j]);
}
e[i]=temp;
//зміни вагових коефіцієнтів
for (int j =layers-1; j>0; j--)
{= (Layer) massivLayers.get (j);= new double[l.getVIxodi()];= l.getMassivOshibok ();= (Layer) massivLayers.get (j-1);= new double [l1.getVIxodi()];= l1.getMassivRas4et();.izmenenieVesov (normaObu4eniya, resultStudy, tempArray);
}
}
//перевірка достовірності=0;(int i = 0; i< epoxa; i++)
{=temp+e[i];
```

```
}=temp/2;(temp<Em) STOP = true;
++;(counter==10)
{JOptionPane.showMessageDialog(null,"Відбулось "+(count2*10)+" проходів навчання");2++; counter=1;}
}
}
```

При реалізації цього методу використовуються об'єкти класу StudyArrays. Кожен об'єкт цього класу – це два масиви, масив вхідних значень та масив цільового виходу. Кількість цих об'єктів відповідає навчальним зразкам і визначається користувачем перед початком процесу навчання нейронної мережі.

3.3. Робота з програмою

Для успішного виконання запропонованої програми необхідно, щоб на ПК був встановлений набір інструментальних засобів Java Runtime Environment. Тоді достатньо переписати до робочої директорії файл NeiroNet.jar і запустити його виконання з командного рядка наступною командою java -jar NeiroNet.jar.

Ми створюватимемо, зберігатимемо і навчатимемо просту нейронну мережу, яка повинна розпізнавати зображення цифри. Цифру представляємо у вигляді бінарного масиву (малюємо за клітинами-пікселями). Темні пікселі відповідають значенню 1 і світлі відповідають значенню 0. Зображення кожної цифри представлено 35 пікселями, це прямокутна матриця 5×7 осередків.

Після запуску програми на екрані з'являється вікно, як показано малюнку 3.1.

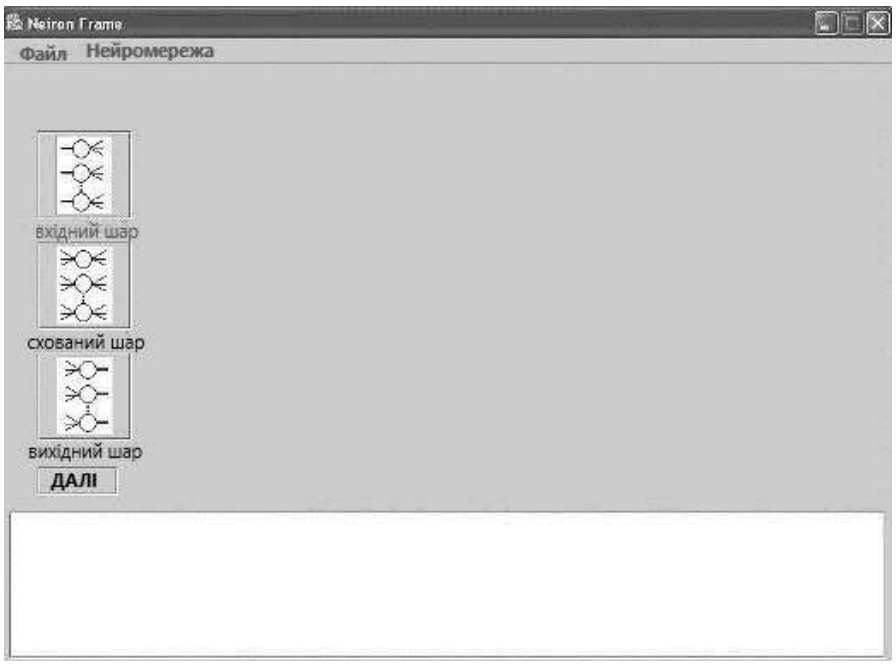

Рисунок 3.1. Вікно вибору шарів мережі.

Користувач, для створення нейронної мережі, повинен відобразити на робочій області вікна всі необхідні йому шари нейронної мережі. Це робиться в такий спосіб. Натискається кнопка відповідна потрібному шару (вхідний, прихований або вихідний).

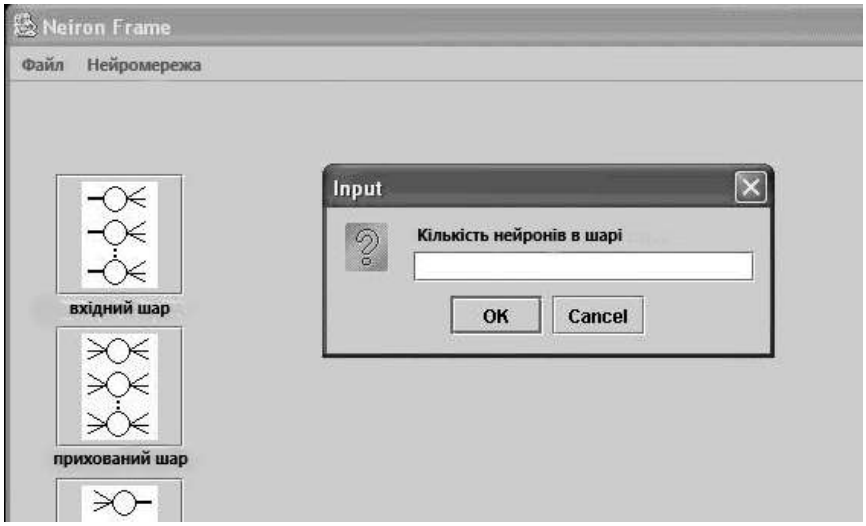

Рисунок 3.2. Ввод количества нейтронніх елементов

Потім клацанням лівої кнопки миші користувач має цей шар. При необхідності зміни розташування зображення шару, в графічному інтерфейсі реалізовано механізм drug-and-drop. Після вказівки розташування шару користувачеві пропонується ввести кількість нейронних елементів у шарі, що створюється, малюнок 4.3. У програмі виключена ситуація створення більше

одного вхідного та більше одного вихідного шару.

Нам потрібно, щоб кожному пікселю відповідав нейронний елемент вхідного шару. Отже, у вхідному шарі буде 35 нейронів. За рекомендаціями щодо навчання та архітектури багатошарових мереж число нейронів у прихованому шарі має бути менше навчальних зразків. Приймемо цю кількість рівну 6. Оскільки цифр може бути 10 (0…9), то для розбиття вхідних образів на 10 вихідних областей треба використовувати 9 нейронів. Для цифри 1 значення на виході першого нейрона буде максимальним, рівним 1, а на всіх інших мінімальним, рівним 0. Відповідно для цифри 2 значення на виході другого нейрона буде максимальним, а на всіх інших мінімальним. І так далі для всіх цифр до 9. Для цифри 0 приймемо значення всіх нейронів вихідного шару рівним нулю.

Отже, маємо три шари. Вхідний з 35 нейронів, прихований, з 6 нейронів та вихідний шар з 9 нейронів. Коли всі ці шари розташовані на робочій ділянці, потрібно натиснути кнопку “Далі”. Після цього створюється модель нейронної мережі. Під час створення об'єкта класу Net буде запит на те, яка функція активації буде використовуватися в нейронах шарів обробки. У нас це бінарна. Після створення модель нейронної мережі стають активними пункти меню, пов'язані з її функціонуванням. Користувач має можливість переглянути властивості мережі, тобто. кількість шарів, кількість нейронів у кожному шарі, та всі вагові коефіцієнти кожного нейронного елемента. Так само з'являється можливість запуску механізмів навчання нейронної мережі, і розрахунку результату на основі даних, що вводяться.

При необхідності зберегти модель нейронної мережі можна вибрати пункт меню “Зберегти як”. Якщо користувачеві необхідно відкрити існуючу модель нейронної мережі, слід вибрати пункт меню “Відкрити”. У діалоговому вікні вибрати шлях до директорії, де зберігається необхідний файл і вибрати його. Після відкриття файлу також стають активними пункти меню для роботи з нейронною мережею.

Навчання нейронної мережі відбувається за принципом навчання з

учителем, згідно з алгоритмом зворотного розповсюдження помилки. Навчання з учителем має на увазі наявність у навчальних зразках не тільки вхідних даних а й точно їм відповідним цільовим значенням вихідних нейронів.

Коли користувач вибирає в меню пункт "Навчання", то в режимі діалогу він вводить усі необхідні дані. Цільові вихідні значення ставляться і виглядають оскільки показано малюнку 3.3.

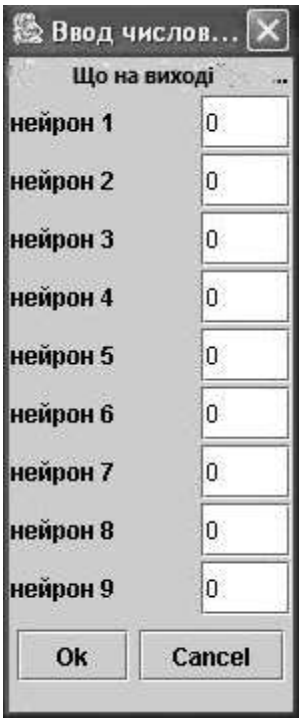

Рисунок 3.3. Вікно введення цільових значень

Після введення останнього зразка програма без додаткових дій користувача починає навчання нейронної мережі. Після закінчення навчання видається повідомлення про завершення процесу.

Якщо ж потрібно перевірити роботу навченої мережі, то слід вибрати в меню пункт “Розрахунок”. Далі нейронна мережа здійснює розрахунок і видає результат На малюнку 4.6 показаний результат розрахунку мережею дослідного зразка для цифри три.

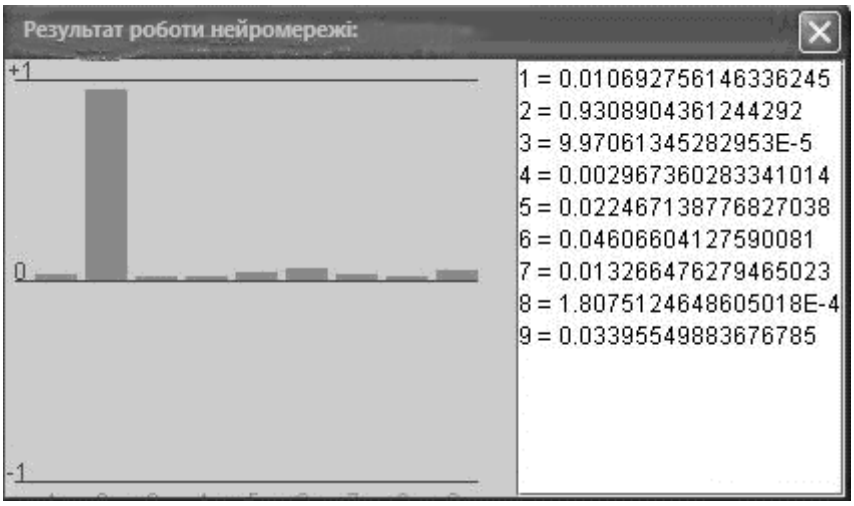

Рисунок 3.4. Виведення результатів

Ми бачимо, що значення на виході третього нейронного елемента є максимально, а всі інші близькі до нуля. Ці значення, максимуми та мінімуми не збігаються з 1 та з 0, оскільки сигмоїдальна функція ніколи не досягає цих значень. Але значення третього вихідного нейронного елемента, що дорівнює 0,93, досить переконливо вказує на те, що на вхід нейронної мережі дійсно подавалося зображення цифри три.

Висновки до третього розділу

В даному розділім було зроблено наступне:

1. Обгрунтовано вибір інструментарію для написання програмної реалізації.

2. Описані етапи розробки класів з яких складається нейронна мережа.

3. Також був реалізований користувацький інтерфейс для демонстрації роботи нейронної мережі

4. Представлена робота з програмою на прикладі створення, налаштування, навчання і розрахунку нейронної мережі на прикладі задачі для розпізнавання цифр.

5. Дана програма дозволяє модифікувати створювану мережу залежно від задач, що будуть поставлені.

..

Загальні висновки

Метою дипломної роботи було вивчення питань проектування нейронних мереж.

Для досягнення поставленої мети було виконано наступні завдання:

1. Проведено аналіз літературних джерел з досліджуваної тематики роботи.
2. Розглядалися основні поняття штучних нейронних мереж, їх класифікація, а також класифікація основних методів навчання. .
3. Досліджено специфіку складу та застосування штучних нейронних мереж.
4. Обрано та обґрунтовано використані засоби розробки.
5. Досліджено найвідоміший алгоритм навчання – так званий алгоритм зворотного поширення (back propagation)..
6. Розроблено діаграму зв’язків та алгоритм роботи штучної нейронної мережі.
7. Розроблено інтерфейс головної форми системи створення штучної нейронної мережі.
8. Виконано моделювання та дослідження процесу тренування розробленої моделі штучної нейронної мережі.

СПИСОК ЛІТЕРАТУРИ

1. Суботін С. О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень : Навчальний посібник / С. О. Суботін. – Запоріжжя: ЗНТУ, 2018. –341 с.

2. Andrew NG. Learn Machine Learning. https://www.coursera.org/learn/machine-learning

3. Adit Deshpande. A Beginner’s Guide To Understanding Convolutional Neural Networks. https://adeshpande3.github.io/

4. Тимощук П.В. Штучні нейронні мережі – Навчальний посібник – Львів: Видавництво Львівська політехніка, 2021. – 444 с.

5. http://www.ipai.net.ua/journal-archive

6. Guido S. Introduction to Machine Learning with Python / S. Guido, A. Müller. – Sebastopol, United States: O'Reilly Media, Inc, USA, 2016. – 392 с

7. Що таке нейронні мережі? Приклади і наші готові рішення. [Електронний ресурс.]:
https://evergreens.com.ua/ua/development-services/neural-network.html

8. The MNIST database [Електронний ресурс]: режим доступу:
http://yann.lecun.com/exdb/mnist/ 9. Нейронні мережі: теорія і практика[Електронний ресурс]: Матеріал з сайту InTalent – режим доступу:
https://intalent.pro/article/neyronnye-setiteoriya-i-praktika.html

10. Машинне навчання[Електронний ресурс]: Матеріал з сайту Machinelearning – режим доступу
http://www.machinelearning/wiki/index.php?title=Машинное_обучение

11. Principles of training multi-layer neural network using backpropagation[Електронний ресурс] – режим доступу:
http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

12. An overview of gradient descent optimization algorithms [Електронний ресурс]: – режим доступу: https://ruder.io/optimizing-gradientdescent/index.html

13. What is OCR and OCR technology[Електронний ресурс]: Матеріал з сайту ABBYY – режим доступу: https://www.abbyy.com/enus/finereader/what-is-ocr/ 9. Офіційний сайт Java[Електронний ресурс]: https://docs.oracle.com/en/java/
14. Машинне навчання простими словами. Електронний ресурс. Код доступу: http://www.mmf.lnu.edu.ua/ar/1739
15. Васенко, Д. В. Методи навчання штучних нейронних мереж. Інформатизація освіти, 2007. - С. 20-29.
16. Krizhevsky A., Sutskever I., Hinton G.E. Imagenet classification with deep convolutional neural networks // Advances in neural information processing systems. - 2012. - С.1097-1105.
17. Generative Adversarial Networks – Hot Topic in Machine Learning. URL : http://www.kdnuggets.com/2017/01/generative-adversarialnetworks-hottopic- machine-learning.html (дата звернення 23.10.2023).
18. ImageNet – Image database for machine learning. URL : http://www.image-net.org/ (дата звернення 23.10.2023).
19. OpenCV library. URL : https://opencv.org/ (дата звернення 23.10.2023).
20. The Most Popular Language For Machine Learning and Data Science Is … URL : http://www.kdnuggets.com/2019/01/most-popularlanguagemachine-learning-data-science.html (дата звернення 23.10.2023).
21.Brownlee J. Loss and Loss Functions for Training Deep Learning Neural Networks, 2019. URL : https://machinelearningmastery.com/loss-andloss-functions-for-training-deep-learning-neural-networks/
22.Brownee J. How to Choose Loss Functions When Training Deep Learning Neural Networks. 2019. URL : https://machinelearningmastery.com/how-to-choose-loss-functions-whentraining-deep-learning-neural-networks/

ДОДАТОК А Окремі фрагменти програмного коду

ДОДАТОК Б ПРЕЗЕНТАЦІЯ